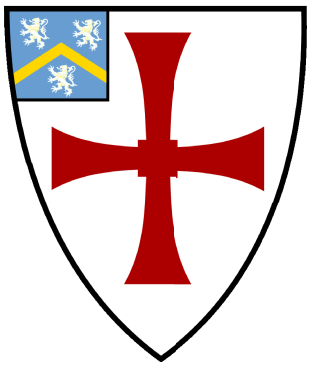
Indexing and Retrieval of 3D Articulated Geometry Models

Gary Kwok-Leung Tam

A Thesis presented for the degree of
Doctor of Philosophy



Department of Computer Science
University of Durham
England

November 2009

Indexing and Retrieval of 3D Articulated Geometry Models

Gary Kwok-Leung Tam

Submitted for the degree of Doctor of Philosophy
2009

Abstract

In this PhD research study, we focus on building a content-based search engine for 3D articulated geometry models. 3D models are essential components in nowadays graphic applications, and are widely used in the game, animation and movies production industry. With the increasing number of these models, a search engine not only provides an entrance to explore such a huge dataset, it also facilitates sharing and reusing among different users. In general, it reduces production costs and time to develop these 3D models. Though a lot of retrieval systems have been proposed in recent years, search engines for 3D articulated geometry models are still in their infancies. Among all the works that we have surveyed, reliability and efficiency are the two main issues that hinder the popularity of such systems. In this research, we have focused our attention mainly to address these two issues.

We have discovered that most existing works design features and matching algorithms in order to reflect the intrinsic properties of these 3D models. For instance, to handle 3D articulated geometry models, it is common to extract skeletons and use graph matching algorithms to compute the similarity. However, since this kind of feature representation is complex, it leads to high complexity of the matching algorithms. As an example, sub-graph isomorphism can be NP-hard for model graph matching. Our solution is based on the understanding that skeletal matching seeks correspondences between the two comparing models. If we can define descriptive features, the correspondence problem can be solved by bag-based matching where fast algorithms are available.

In the first part of the research, we propose a feature extraction algorithm to extract such descriptive features. We then convert the skeletal matching problems into bag-based matching. We further define metric similarity measure so as to support fast search. We demonstrate the advantages of this idea in our experiments. The improvement on precision is 12% better at high recall. The indexing search of 3D model is 24 times faster than the state of the art if only the first relevant result is returned. However, improving the quality of descriptive features pays the price of high dimensionality. Curse of dimensionality is a notorious problem on large multimedia databases. The computation time scales exponentially as the dimension increases, and indexing techniques may not be useful in such situation.

In the second part of the research, we focus ourselves on developing an embedding retrieval framework to solve the high dimensionality problem. We first argue that our proposed matching method projects 3D models on manifolds. We then use manifold learning technique to reduce dimensionality and maximize intra-class distances. We further propose a numerical method to sub-sample and fast search databases. To preserve retrieval accuracy using fewer landmark objects, we propose an alignment method which is also beneficial to existing works for fast search. The advantages of the retrieval framework are demonstrated in our experiments that it alleviates the problem of curse of dimensionality. It also improves the efficiency (3.4 times faster) and accuracy (30% more accurate) of our matching algorithm proposed above.

In the third part of the research, we also study a closely related area, 3D motions. 3D motions are captured by sticking sensor on human beings. These captured data are real human motions that are used to animate 3D articulated geometry models. Creating realistic 3D motions is an expensive and tedious task. Although 3D motions are very different from 3D articulated geometry models, we observe that existing works also suffer from the problem of temporal structure matching. This also leads to low efficiency in the matching algorithms. We apply the same idea of bag-based matching into the work of 3D motions. From our experiments, the proposed method has a 13% improvement on precision at high recall and is 12 times faster than existing works.

As a summary, we have developed algorithms for 3D articulated geometry models

and 3D motions, covering feature extraction, feature matching, indexing and fast search methods. Through various experiments, our idea of converting restricted matching to bag-based matching improves matching efficiency and reliability. These have been shown in both 3D articulated geometry models and 3D motions. We have also connected 3D matching to the area of manifold learning. The embedding retrieval framework not only improves efficiency and accuracy, but has also opened a new area of research.

Declaration

The work in this thesis is based on research carried out in the Department of Computer Sciences, University of Durham, England. No part of this thesis has been submitted elsewhere for any other degree or qualification and it is all my own work unless referenced to the contrary in the text. The research work on matching and retrieval of motion sequences is a joint effort with Mr. Mark Corbyn and Mr. Qingzheng Zheng. My contribution is to research and propose a new feature extraction and matching method for 3D motions. Mr. Mark Corbyn and Mr. Qingzheng Zheng implemented my proposed method and other related works for comparison.

Acknowledgements

I would like to extend my appreciation and gratitude to the following people for their contributions to this research study.

With my deepest gratitude and respect, I am especially indebted to my supervisor Prof. Rynson W.H. Lau. His insightful remarks and guidance have expanded my horizons in my research study. This thesis would not be completed without his invaluable advice, patience and endless support. I would also like to thank the two thesis examiners, Prof. Min Chen and Dr Ioannis Ivrissimtzis for their insightful and constructive comments on the thesis.

Special thanks go to Dr. Frederick W.B. Li, who has given me support, insightful comments and care since my supervisor has left Durham. Many thanks also go to Mr. Qingzheng Zheng, Mr Addison S.K. Chan, Mr. Fabio A. Domingos, Dr. Cole J. Guo, Dr. Yonghong Xiang, Mr. Pim van 't Hof, Ms. Barbara Froner, Mr Khalid Mohammad, Mr. Xiaofeng Du and Dr M.J. Sun for their great friendship. I would also like to thank Mr. Mark Corbyn for the collaboration.

Most of all, my wholehearted thanks go to my mother and sister for their unconditional love and support. And most importantly, my deepest thanks must go to my dearest wife, Dr Laureen L.Y. Chan, for her love, companion, encouragement, concern, understanding and inspiration.

Table of Contents

List of Figures

List of Tables

Chapter 1

Introduction

3D geometry models and 3D motions are essential components in latest graphics applications. Ranging from 3D games, animations and movies, they define the objects and actions that we see in the virtual world. 3D articulated geometry models are a subset of 3D models that are articulated into different postures, e.g., a sitting dog and a running dog. They are both dog objects but in different poses. 3D motions are recorded time-series data that are captured from real human motions using motion sensor stuck on human bodies. Example motions include running and jumping. With the increasing number of these data, recent research has been focusing on building a reliable search engine so as to facilitate share and reuse.

1.1 Motivation

The urge for such retrieval system comes from the fact that these 3D models and motions are difficult, time-consuming and expensive to create. We can see these in several ways. Traditionally, one has to use 3D modeling tools (e.g., 3D Studio Max, Maya) to handle such 3D models. Not to mention the high cost of the software licenses, there is also a high demand on the expensive graphic hardware. A user typically relies on the provided GUI and mouse to manipulate models in a 2D display. Since we all live in a 3D world and there is 1 fewer degree of freedom (2D) of control, the manipulations are usually different from what a general user would expect. Sometimes, a user may even feel dizzy because of the lack of virtual

3D perception on a 2D screen. Adding on top, a user must have knowledge on geometry and topology in order to produce high quality models. The same applies to 3D motions as well. Building skeletal structures, animating muscles and limbs, acquiring the deep knowledge of human motions and the traits of the animating characters are some of the challenges that an animator must face. Despite the steep learning curve of modeling and animating, our human visual system highly adapts to real-life activities and so we can easily recognize any subtle differences and abnormality. This, on the other hand, creates a high demand to produce very realistic models and motions.

To facilitate creation of such 3D contents, the concept of "copying and pasting" existing components has recently been introduced. Imagining creating a centaur model, it would be much easier to copy the upper part of a human model and the lower part of a horse model and paste them together. It is called "Modeling by Example" [1] in 3D modelling literature. Similarly, imagine creating a triple jump motion, concatenating various running, stepping and jumping motions would greatly ease the animator's job [2], [3]. However, all these require efficient and reliable search engines.

Apart from content creation, as the popularity of graphics applications increases, the growing number of these models and motions also creates other problems as well. In term of storage, the huge volume of data easily grows over gigabytes or terabytes. Though the capacity of hard drive is increasing, it is hardly able to catch up with the demand. Traditional solution is to put them into slow removal secondary storage like tapes and DVDs. It is well-known that the indexing structure of a retrieval system is always useful to reduce slow disk access during such multimedia search. In term of searching, browsing through pages and pages of a repository for the appropriate content satisfying certain requirements becomes more and more difficult also. All these motivate us to research reliable and efficient retrieval methods for these 3D contents.

1.2 Content-based Retrieval

Traditional retrieval methods are based on annotations. Examples include audio, image, video and text retrieval systems. However, human annotation depends on many factors, such as languages, cultures, personal experiences and even the ways that the contents are recognized or used. There is no definite answer for manual annotation. Recently, research has been focused on content-based retrieval techniques. Content-based methods analyze the multimedia data from their actual contents. Features are then extracted and compared automatically without manual interference. One of the goals of content-based methods is to define a robust and accurate measure to assist automatic matching, recognition and classification or even generate reliable annotations.

Content-based techniques, in general, comprise three main parts: feature extraction, feature matching and fast search. Feature extraction concerns the extraction of features which can be used to represent the data. Feature matching concerns the comparison of features and is always accompanied by a distance function for measuring the similarity between two features. In order to allow fast online query search, an indexing technique is often applied. It provides an efficient method to support fast pruning of irrelevant data.

1.3 Project Objectives

In the research, we devote ourselves to the development of content-based retrieval techniques for 3D articulated geometry models, focusing on efficiency and reliability. They are the main issues that hinder the popularity and practicality of such retrieval systems. The inputs, 3D articulated geometry models, are expressed as 3D meshes. Each model is simply a collection of vertices, edges and triangles without any tags or skeletal information. Since our methods are content-based, the system analyzes the query input, extracts descriptive features and searches for relevant data using the predefined matching methods. Since 3D articulated geometry models are usually animated by 3D motion capture data, we also devote our time to develop a content-based retrieval technique for motions. Similarly, these 3D motions are time-series

data without any annotations or tags.

1.4 Problems

Though a lot of works have been proposed in recent years, retrievals of 3D articulated geometry models and motions are still in their infancies. From the literatures, we have observed some limiting factors.

Firstly, most existing works designed features and matching methods in order to reflect the intrinsic properties. For instance, to handle 3D articulated geometry models, it was common to extract the skeleton and use sub-graph matching (e.g. [4], [5]) to define similarity measures. To handle 3D motion capture data, special focus was put on temporal structure matching. In order to handle such special data represetation, matching algorithms usually have high complexity, and so special indexing (e.g. [6], [7]) or bounding techniques (e.g. [8], [9]) are required.

Secondly, the scalability issue is another area that has not been explored. On the one hand, many works [4], [5], [10], [11], [12], [13], [14], [15], [16], [13] define non-metric similarity measures where general efficient indexing techniques (e.g. spatial or distance-based indexing (see Section 2.4)) cannot be applied. On the other hand, when the methods define large number of features [17], they will easily suffer from the curse of dimensionality and huge computation time is required to handle such datasets (see Discussion and Experiments in Section 5.6).

1.5 Proposed Solution

Similar to existing works, we also consider extracting features that reflect the intrinsic properties of the data. However, we also propose to convert the restricted matching problem into bag-based matching problems. Our argument is that one of the goals of these matching methods is to seek correspondences between the two comparing models. If we can define descriptive features, correspondence problems can be solved by bag-based matching where efficient algorithms are available. Therefore, in the first part of the research, we devote ourselves into the analysis of feature

extraction and feature matching of 3D articulated geometry models. Our focus is to develop a metric similarity measure so that distance-based indexing techniques can be applied.

To alleviate the problem of dimensionality, we propose an embedding retrieval framework for fast search. In particular, we focus ourselves on the matching of 3D articulated geometry models. We discover that graph and bag-based matching algorithms project data on manifolds in high dimensional space. These trigger us to investigate the reason behind and to propose a manifold learning method for dimension reduction and inter-class distance maximization. To extend the framework to a large dataset, we adapt Nyström extension (a sub-sampling scheme) [18] and further propose an alignment step to preserve retrieval accuracy.

While 3D motions are different from 3D articulated geometry models, they suffer from similar limiting factors. Similarly, we also propose to convert temporal matching into bag-based matching by defining proper and descriptive features. This results in a metric similarity measure that is capable for fast search.

1.6 Contributions

In brief, three contributions are achieved in this PhD research study.

1. We develop a feature extraction and matching method for 3D articulated geometry models. It is an improvement on our earlier work [19]. The improvements are as follows. 1) we have developed a more reliable feature extraction algorithm that solves the slicing direction problem. We name our method "Topological Point Ring (TPR) Analysis" as it uses topological points and rings as features. 2) By properly adjusting the importance of these topological features, we have defined the first metric similarity measure which allows both skeletal and geometry feature matching at the same time. Since it is a metric, it also supports the use of indexing technique for fast pruning. The method is also faster and more accurate than the state of the art, Multiresolution Reeb Graph (MRG) [17]. These improvements and findings are reported in a paper [20] in the Transactions on Visualization and Computer Graphics.

2. To further improve the reliability and efficiency of the proposed retrieval method for 3D articulated geometry models, we investigate and propose an embedding retrieval scheme for fast search. When applying our framework on TPR [20] and MRG [17], improvements on efficiency and accuracy are reported in our experimental results. These findings are currently under review for publication [21].

3. Applying the concept of bag-based matching, we have also developed a feature extraction and a feature matching algorithm for 3D motions, featuring local and global matching at the same time. The method is also shown to be faster and more accurate than two existing works, namely Dynamic Time Warping [8] and Uniform Scaling [9]. These results have been published in a conference paper [22].

All in all, by converting restricted (skeletal and temporal) matching into bag-based matching, we have developed new method for the analysis of 3D articulated geometry models and 3D motions. Our investigation into the existing work also discovers that graph-based and bag-based matching algorithms project models on manifolds. This connects the area to manifold learning techniques.

1.7 Organization

The rest of the thesis is organized as follows. In Chapter 2, we survey existing matching and retrieval work of 3D articulated geometry models and various fast search and indexing schemes. We then discuss our research goals, the challenges and the proposed solutions in Chapter 3, In Chapter 4, we develop a feature extraction and matching method for 3D articulated geometry models, featuring a metric similarity measure that supports indexing. In Chapter 5, we investigates an embedding retrieval framework to improve speed and accuracy for the matching of 3D articulated geometry models. In Chapter 6, we apply the concept of bag-based matching to the retrieval of 3D motion capture data, featuring a metric similarity measure. Since it is a piece of independent work, we present the surveys, proposed retrieval method and experimental results in a self-contained manner in Chapter 6. Finally,

in Chapters 7 and 8, we evaluate the whole research study, draw our conclusions and discuss some future works.

Chapter 2

Literature Review

2.1 Introduction

In this chapter, we survey and discuss works related to this research study. In Section 2.2, we first discuss some general 3D models retrieval methods. These works cannot handle articulated models, but provide the background of our current works. In Section 2.3, we discuss matching methods for articulated geometry models. Since this is the focus of our work, we discuss them in detail. In Section 2.4, we discuss general fast search methods. This provides the background of our proposed embedding retrieval framework.

2.2 Methods for Non-Articulated Geometry Models

There is a substantial amount of work devoted to matching and retrieving rigid geometry models efficiently and accurately. These non-articulated methods can be classified into three approaches: geometry-based, transform-based, and image-based approaches. The geometry-based approach concerns properties related to the shape and size of a model. In general, methods of this approach can be classified into three types: methods based on extracting physical properties [23], [24], [25], methods based on computing histograms or some distribution functions [26], [27],

[28], and methods based on computing energy for morphing a model [29], [30], [31]. The transform-based approach analyzes 3D models in a different feature domain. Transformation functions include Fourier Transform [32], Wavelets Transform [33], and Zernike Transform [34]. Funkhouser et al. [35], Kazhdan et al. [36], and Novotni and Klein [34] propose Spherical Harmonic for extracting rotation-invariant features. The image-based approach captures features from different 2D image views of a 3D model [37], [38].

Generally, the geometry-based approach is efficient and easy to implement, but its matching accuracy is usually lower than the other two approaches. The transform-based approach has several advantages such as supporting multiresolution analysis and having improved accuracy with the recent development in concentric spherical harmonic [36]. A major advantage of the image-based approach is its independence from 3D data representation. However, it typically has a large number of features and, hence, high matching cost. It should be noted that the image-based approach can give a better human-computer interface [39] because users may provide a 2D sketch as an initial query input.

All these methods, however, are designed to handle general 3D models (e.g., chairs, tables) only. When a model (e.g., boy) undergoes large articulation changes (e.g., in the form of crawling and running), all these methods will consider them totally different models. The reason is that these methods rely on properties (e.g., reflective symmetry plane, anisotropy, center of mass, rotation axis) that are not invariant under articulation changes. To handle articulation, special methods have to be developed.

2.3 Methods for Articulated Geometry Models

The analysis of 3D articulated geometry models stresses on the capability to extract articulation invariant features. In term of feature extraction, topological invariant properties and skeletons are the two most important and frequently used properties. Theoretically, topology studies properties of mathematical structure such as connectedness, continuity etc. When applied on 3D shape analysis, the major focus

is to find and use properties that exhitbit topological invariance, like isometric deformation and bending without tearing the 3D shapes. Some of these concepts and properties include: geodesic, Morse theory, reeb graph [17], level set [40], size function [41] etc. Another important property in 3D shape analysis is skeleton, which has two major definitions. One uses the Medial Axis Transformation (MAT) [42]. It represents the locus of points that are equi-distances from the surface boundary. Note that in 3D, the MAT representation is usually not 1 dimensional. However, with a proper distance function measuring the distance from MAT to the surface, the whole surface can be easily reconstructed using the MAT representation. Another important defintion of skeletons roots from the Morse Theory. The theory studies surfaces by defining critical points where the derivative of a scalar function of a point is zero. By connecting these critical points, Level Set Diagram (LSD) [40], a form of 1D skeleton can be built. Apart from that, by dividing a surface into different components with respect to a height function, a reeb graph can be built. Reeb graph can also be considered a type of skeletal representation [17, 41].

As we can see, these two properties are related in some sense, and all surveyed methods make use of these properties. To better understand the differences and performances of these works, we categorize these methods into three approaches. They are *Bag of features*, *Single feature vector* and *Pose-normalization* approach. Since they are the focuses of this research, we discuss these methods in detail.

2.3.1 Bag of Features

There are lots of works based on this approach. These methods use a bag of features (scalars or vectors) to represent each 3D articulated geometry model. We further classify these works into Graph-Based Methods and Bag-Based Methods.

2.3.1.1 Graph-Based Methods

Graph-based methods partition a model based on some metrics on the surface. Features are then extracted from each of these partitions and stored in a graph structure. These features are related to each others in the graph as parents, children

and siblings. The whole graph represents a model. To define a similarity measure, a graph matching algorithm is employed.

Tal et al. [5] analyze models based on component graph. It first segments a model based on a mesh decomposition algorithm. Each component node is then fitted by one primitive. The choice of primitive is based on a non-linear least-square optimization algorithm. All these primitives are then connected to neighboring components to form a component graph. To match two component graphs, an optimal error-correcting sub-graph isomorphism algorithm is applied.

In [4], Sunder et al. apply voxel thinning to extract skeleton from a voxelized model. A clustering algorithm is then applied to extract nodes for constructing a skeletal graph. Apart from skeletal matching, the radial distribution of the node edges is preserved for local shape matching. To match two skeletons, a recursive bipartite algorithm is applied. Such algorithm is enhanced by a greedy depth-first search so that the matching follows the skeletal structure. The work also uses a graph indexing technique for fast skeleton pruning and then considers the distribution of node edges to further prune irrelevant models. Graph indexing is also applied for fast pruning in [6] and [7], where the former uses Laplacian as the indexing spectrum, and the later considers both spectrum and geometric features (primitive number) in the indexing process.

Apart from skeleton, Hilaga et al. [17] introduce the Multiresolution Reeb Graph (MRG) to represent a model. It first partitions a model into different intervals using integral geodesic (or average geodesic, centricity in some other works). Integral geodesic measures how far a point is away from the surface center. Unlike Euclidean distance, geodesic distance [43, 44] is measured on the model surface and is not affected by model deformation. Then, by analyzing the parent, siblings and children of each component in each interval, a Multi-Resolution Reeb Graph is built in a hierarchical manner. In each node, area and length are used as geometric features. To match two MRG trees, a heuristic algorithm is applied in a coarse-to-fine manner. It first matches the two root nodes in the two MRG trees. Then the matching traverses down the two MRG trees following the child nodes with maximum similarity. The matching goes on until all the nodes in either MRG trees are matched. The work

has two advantages: multiresolution support and the flexibility to match additional geometric features.

The success and flexibility of MRG has also stimulated many other works. For example, Tung et al. [10] propose to use additional geometric features to improve matching accuracy. The idea of reeb graph has also been extended in [45] together with inexact subgraph-isomorphism to produce a subpart matching method. Inspired by MRG, Bespalov et al. recursively subdivide a model into surface patches and store features in a binary tree using Scale-Space decomposition. The decomposition is based on singular value decomposition on the matrix built from two distance functions defined on the surfaces. These distance functions are geodesic distance [11] and maximum angle on angular shortest path [12]. The similarity measure is then defined by sub-part correspondence. Node matching is defined similar to [17] but is used in a subgraph-isomorphism matching algorithm.

2.3.1.2 Bag-Based Methods

Apart from graphs, other methods consider model signature as a bag of features instead. These features (within a bag) are all unrelated to each other. This differs from those in graph-based methods and so it allows strayed matching (e.g., a finger can be matched to a leg). The similarity measure between two bags is then defined by matching algorithm which targets at finding correspondences. The idea is that if the geometric features are distinctive enough, model matching is equivalent to finding the best corresponding match between individual features. This discourages strayed matching. Since these methods consider model signature as a set (bag), we term all these methods as bag-based matching methods.

Tam et al. [46] capture topologically important features points at protrusions and joints of a 3D model. It then associates curvature histogram to each of these points. These histograms are articulation invariant because they follow the geodesic distance on the surface. Tierny et al. [13] define patch signatures using annulus-like chart unfolding algorithm. Such patch signature measures the stress of unfolding. Instead of capturing distinctive features, Ruggeri et al. [14] use lots of evenly distributed points and point histograms for matching. All these methods apply the bipartite

algorithm to find correspondence and match the two sets of features.

Apart from bipartite matching, other works use Earth Mover Distance instead. Earth Mover Distance is frequently used in image retrieval, and provides a similarity measure that is closer to human perception [47]. It computes the minimal energy required to morph from one feature set to another. It also allows the association of a different importance value to each feature in the set. In [15], Tam et al. segment a model into regions and define importance based on bounded regions. It then uses curvature and area histogram as features. The dissimilarity measure is then defined by Earth Mover Distance between two feature sets. Instead of using compact features, Cornea et al. [16] capture skeleton from a voxelized model. The skeleton, which is represented as a set of voxels, is then morphed to another skeleton. The voxel-to-voxel morphing is carried out using Earth Mover Distance directly. [16], [13], [14] show that they can handle subpart matching as well.

Biasotti et al. [41] propose a size function to represent a 3D model. Size function is a mathematical tool that is used for image retrieval and classification. It maps each pair of topological entity and measuring function (geometric features) into a 2D point. The similarity measure is then defined by finite point set matching. The method has several advantages, e.g., metric properties, robust to noise and flexible to various shape and measuring functions.

2.3.2 Single Feature Vector

This approach uses a single feature vector (called shape descriptor) to represent the whole model. Many of these methods construct histograms based on metrics defined on the surface.

In [48] a 2D shape descriptor is proposed. The descriptor is a combination of the distribution of two scalar functions: local diameter and centricity function. Local diameter function measures the thickness of the 3D shape in the neighborhood of each vertex. The centricity function measures the average geodesic distance from a vertex to all other vertices on the mesh. The first function provides descriptive shape information where the second provides the spatial information.

In [49] a 3D eccentricity transform is proposed which is an extension of its 2D

case. For each point p, it assigns the maximum geodesic distance of the whole mesh from p. Such transform is shown to be invariant to articulation and noise. The transform is more robust than centricity but it requires voxelization as a pre-processing step.

In [50] a part-aware metric measure is proposed. Part-aware metric is a combination of a volumetric shape image (VSI), geodesic distance and normal variation. Volumetric shape image quantifies the visual region that is seen from a point. These visual regions correspond to parts (convex regions) of a model, and provide a descriptive measure especially for parts and shapes.

Apart from seeking various metrics defined on the surface, Reuter et al. [51] propose to use spectrum (leading eigenvalues) of the Laplace-Beltrami operator computed on the surface as descriptor. These leading eigenvalues corresponds to the significant components (i.e., structure) of a surface. Comparing two spectrums is thus similar to comparing two model structures.

2.3.3 Pose-normalization

Though most works focus on capturing articulation invariant features, some researchers strive for a more challenging task: normalizing the pose.

In [52], Multi-Dimensional Scaling (MDS) [53] was used for pose-normalization. Given a distance matrix, MDS is a numerical tool to find an N-dimensional embedding space that best preserves all (close and far) distances using stress minimization. When pairwise geodesic distances of every point on a surface are projected into such low-dimensional embedding, Elad et al. observe that the resulting isometric surface is bending-invariant (normalized). This allows general 3D model matching algorithms to be applied on the normalized model directly.

Similarly, Jain et al. [54] propose to use geodesic affinity matrix together with a kernel method to obtain a spectral embedding. Using a kernel method means that such method preserves local distances. This is desirable because a surface is a manifold which is defined by local neighborhood. Spectral embedding is shown to be more reliable than MDS in the experiments.

Rustamov [55] considers a similar idea by embedding every point of a model

through the eigenvectors of the associated Laplace-Beltrami operator. Though such embedded vertices cannot be visualized as in the previous two methods, the embedded data also forms a surface. To compare the similarity between these surfaces, they use a distribution method [27] which is a general 3D model matching algorithm (Section 2.2). By doing so, the method can avoid the orientation problem.

2.3.4 Summary and Discussion

After reviewing existing matching work of 3D articulated geometry models, we give a discussion on all these methods in this section.

The *Pose-normalization* approach embeds a model into a transformed space so that the posture of the model becomes normalized. It allows the application of general 3D matching methods onto the analysis of articulated geometry models. There are a lot of successful works discussed in Section 2.2. However, since the embedding is obtained from a few top eigenvectors, it leads to the loss of most geometry details which are stored in the rest of the eigenvectors. This suggests that they are useful for matching structure, but may require additional efforts for matching in detail.

The *Single feature vector* approach provides the simplest and fastest matching method among all. First, the feature representation is compact (a 1D/2D vector). Second, the use of Minkowski distance (L_p-norm) means that the computation of similarity measure is fast. In fact, most of the latest research work focuses on defining mesh signature. These signatures are not only useful for comparing shapes, but are also useful for mesh analysis. For instance, the local diameter (thickness) function [48], and part-aware metric [50] have been demonstrated to be useful for part segmentation. Since these methods produce single feature vectors, general spatial indexing techniques (Section 2.4.1.1) can be applied.

Despite all these advantages, there are drawbacks of this approach. First these methods use one single feature vector to represent the whole model. Though it is compact, it may not be descriptive enough to discriminate highly similar skeleton models (e.g., dog and wolf). For example, spectrum [51] may be useful for comparing structure, but it is well-known that, like Fourier Transform, the leading spectrum

usually encode smooth signal but not detail information. Second, it has been argued that simple feature vector and the use of L_p-norm do not always represent the human perception properly [56], [57]. Therefore, striving for distinct mesh signature may not help improve the accuracy of such retrieval system.

On the other hand, it has been argued that similarity between two shapes are contributed by similarity of individual parts [58] in the area of cognition psychology. Breaking a 3D model into smaller subparts and defining similarity measure based on coherence of subparts are the major ideas of the *Bag of features* approach. Graph or Bag-based matching find the correspondences in individual parts and compute similarities accordingly. This suggests that they usually provide similarity measures that better reflect human perception [47]. In fact, as shown in our survey, many of the graph ([45]) and bag-based ([16], [13], [14]) matching algorithms can be extended or used for partial (subpart) matching. Another advantage of the *Bag of features* approach is that it incorporates many features which are essential to distinguish highly similar skeleton models.

Though *Bag of features* approach seems promising, the complexity and efficiency of the matching algorithm hinder the popularity of these methods. As pointed out in [4, 59], if no heuristic algorithm can be found, skeletal graph matching usually suffers from sub-graph isomorphism which is NP-hard. Apart from this, most of the similarity measures provided by these graph or bag-based matching algorithms are all non-metric, suggesting that the scalability of these algorithms has not been explored in large databases.

To solve the first problem, Tam et al. in [46] and [15] proposes to reduce feature sizes and convert graph matching into bag-based matching. To scale slow graph matching to large database, the general approach is to index models by graph (skeleton) spectrum [4, 6]). This is a two-step process. First, graph spectrum is used to fast prune irrelevant models with totally different skeletal structures. The second step applies (slow) graph matching using geometric features to calculate similarity scores. However, such approach may still suffer from the efficiency problem as it separates topological matching and geometric matching into two processes. Considering a database containing many highly similar skeleton models (e.g., dog, wolf, cat,

lion, horse), the first pruning step will not be effective because the graph (skeletal) spectrums of all these models would be the same. Due to this reason, [7] suggests to incorporate geometric features in the graph spectrum. Biasotti et al. [41] also propose a metric size function for similarity measure which suggests that distance-based indexing can be used. However, both the geometric features used in these two methods (a number representing primitive type) [7] and (a single measuring function) [41] are relatively simple, and so may not be useful for discriminating highly similar skeleton models.

2.4 Indexing and Fast Search Methods

Our focus of this research is to develop a fast search technique for the retrieval of 3D articulated geometry models. Since the area is still in its infancy, not much work has been developed to support fast searching. We have covered most of these methods in the previous section. In this section, we survey general fast search techniques. By studying these methods, we can better equip ourselves and develop our fast search scheme that is suitable for 3D articulated geometry models. It should be noted that, nearest neighbor search has been an on-going research for decades. A complete survey is beyond the scope of this research. We roughly group these methods into two main areas: *Metric* approach and *Non-metric* approach.

2.4.1 The Metric Approach

The Metric approach assumes that a metric distance is available. They can be further subdivided into two types: spatial indexing and distance-based indexing. Since the similarity measure (distance) is a metric, all of them make use of triangle inequality for fast pruning. Practically, spatial indexing operates on a vector space and it assumes a vector coordinate (e.g., $< a, b, c >$ is a coordinate in 3D space) whilst distance-based indexing operates on a metric space (e.g., a distance measure that follows metric properties). Mathematically, vector space is a subspace of metric space, because all l_p-norms are metrics.

Spatial (Section 2.4.1.1) and Distance-based indexing (Section 2.4.1.2) are tech-

niques that give exact solutions. However, all these works suffer from the curse of dimensionality problem. We also discuss some techniques that better handle the problem in Section 2.4.1.3, but they result in approximate solutions.

2.4.1.1 Spatial Indexing

A large amount of work that targets at nearest neighbor retrieval in multidimensional vector space has been developed in the past decades. A more comprehensive survey can be found in [60], [61] and [62]. These methods employ Euclidean distance (L_2-norm) as the similarity measure. They first partition the data vector space recursively, according to the data variances on the axes, and represent each partition as a node of an indexing tree. Kd-Tree and R-Tree are some of the notable examples in the area.

2.4.1.2 Distance-based Indexing

Apart from multidimensional vector space, other methods operate on a more general space, the metric space. These methods relax the requirement of vector coordinate representations. The only requirement [62] is a metric similarity measure. Hausdorff Distance [63] and size function [41] (see Section 2.3.4) are some of the examples of metric distances. These algorithms also recursively and hierarchically partition the database to form indexing trees. The major difference is that each node is a partition based on a threshold distance to a pivot object (not the data spread on axes). Some notable examples include VP-tree [64], MVP-tree [65] and M-tree [66, 67].

2.4.1.3 Curse of Dimensionality

Spatial and distance-based indexing work well when the dimension is small. When the number of dimensions increases, these methods soon approach brute-force. It has been argued that as the dimension increases, it is ineffective to partition the vector or metric space. This means that large portions of the database have to be inspected during search. Such effect has been termed "Curse of Dimensionality". Formally, it defines that the complexity of nearest neighbor search scales exponentially with dimension. It has also been proven that curse of dimensionality is

inevitable when dimension exceeds certain threshold [68]. This leads researchers to consider algorithms that allow approximate solutions.

Locality Sensitive Hashing (LSH) [69], an approximate nearest neighbor method, has been proposed. The method hashes similar items in the same bin, and is shown to scale well with high dimensionality theoretically and practically. The method also provides the basis for many successful variants in recent years. However, it can only be applied in the space of l_p-norm.

Apart from LSH, lots of method are proposed using dimension reduction. These methods embed pairwise distances into a vector space. Some of these methods, in particular, Multidimensional Scaling (MDS) [53], Bourgain embeddings [70] need to evaluate exact distances between the query and most of the database objects and, thus, are not designed for efficient nearest neighbor retrieval. Methods that can support efficient search include Lipschitz embedding [70], Sparse Map [71], Fastmap [72], Metric Map [73] and Landmark MDS [74]. It should be noted that Fastmap, Metric Map and Landmark MDS are all variants of Nyström extension [75]. Though these methods try to preserve a large amount of the proximity structure (close and far distances) by stress minimization in the original space, false dismissals are unavoidable. False dismissal refers to the situation that relevant objects exists in the database but are absent in the query result.

Though dimensionality imposes great difficulties (curses), it also provides blessings to data analysis [76]. In [77], Korn et al. show that if the data possess self-similarity and lies on a low dimensional manifold, the complexity of search will depend only on the intrinsic dimensionality of the manifold. These stimulate a lot of works in manifold learning. Some the notable examples include Locally Linear Embedding (LLE) [78] and Isomap [79].

2.4.2 The Non-Metric Approach

There are quite a lot of similarity measures that are non-metric. In non-metric spaces, the triangle inequality does not hold, so it is very challenging to develop efficient similarity search. As human perception is found to be a non-metric measure [80], we would expect that non-metric similarities are very useful. In fact, in Section

2.3, most of the graph and bag-based matching methods are non-metric. In this subsection, we give a brief review on fast search methods that can support non-metric measures. These methods can be further classified into two types of methods: Exact methods, and Approximate methods. Exact methods do not introduce false dismissal while approximate methods do.

2.4.2.1 Exact Method

There are a few general exact methods available. Constant Shift Embedding [81] and Local Constant Embedding [80] are two notable works. The idea of [81] is to convert a non-metric distance into a metric one, assuming that the query is in the database. The conversion is made by adding a very large constant value to the violated triangle inequality so that it follows triangle inequality in the new converted distance. However, the large constant leads to small lower bound which is not useful for fast pruning. [80] improves the concept by introducing various local constants on different groups and allowing dynamic query. The algorithm however requires grouping and searching across different groups.

Apart from these, specific distance measures may be sped up by a technique call filter-and-refine. Similarity measures in time series database is one of such areas. Keogh et al. propose in [8] and [82] to use approximation for lower bounding. After fast filtering by these approximations, exact matching is applied to find the best match. The similarity measures in concerns include Longest Common Subsequence and Dynamic Time Warping, which are both non-metric. However, these filter-and-refine methods are usually constructed particularly for specific distance measures only, and are not applicable to arbitrary distance functions.

2.4.2.2 Approximate Method

Since the exact method is a challenging topic, many works solve a relaxed retrieval problem by allowing false dismissal. In [83], Skopal uses a class of metric-preserving and similarity-preserving modifiers to convert a non-metric similarity measure into a metric one. However, not every non-metric measure is convertible as pointed out by the authors. Athitsos et al. propose an approximate method to map data objects

into vector space by combining several weak classifiers in [84] and [85]. However, requiring lots of classifiers may be difficult with respect to the data in concern.

Similar to metric approach, it is possible to transform non-metric distances into distances of vector space via embedding. However, doing so will again lead to false dismissal. These methods share similar techniques as discussed in Section 2.4.1.3. Apart from embedding methods, some methods convert the non-metric retrieval problem into a classification problem. These methods involve three steps: 1) clustering data using non-metric distance, 2) selecting representative objects in each cluster, and 3) applying similarity search to classify a query object. The data in the identified class become query results. Among all, representative objects used include atypical objects [56] and correlated objects [57].

2.4.3 Our Approach

In this work, we first derive a metric measure based on Earth Mover Distance. From this, we can make use of distance-based indexing technique (Section 2.4.1.2), VP-Tree, for fast retrieval. However, as we later show, such method still suffers from curse of dimensionality.

In our second attempt, we find that using manifold learning and dimension reduction technique, Diffusion Map, can help alleviate the problem. Our method can be considered most similar to Locally Linear Embedding (LLE) [78] and Isomap [79], and is directly related to Nyström-based [75] retrieval methods (Section 2.4.1.3). Since we embed data points into Euclidean space, we would then use spatial-indexing technique, kd-tree, for fast retrieval (Section 2.4.1.1).

Chapter 3

Research Goals, Challenges and Proposed Solutions

3.1 Research Goals

As explained in previous chapters, research for a successful retrieval system of 3D articulated geometry models can be beneficial to many graphics applications, especially game and movie production. However, as shown in our literature review, every approach has its advantages and disadvantages. Since the area is still in its infancy, there are still a lot of directions unexplored. Due to the limited time and resources, we believe that defining research goals as guidance would be advantageous.

We hope that our retrieval method can satisfy the following research goals:

1. **Focus on 3D articulated geometry models** There are many 3D formats: ill-defined 3D models (polygon soup or point cloud models), (non)-manifold meshes, (non)-closed meshes, manufacturing models. We would like to focus on 3D articulated geometry models which are represented by closed, manifold and triangulated meshes. These meshes are collections of vertices, edges and triangles, without any skeletal information provided.

2. **Content-based technique** We would like to develop a retrieval system that is based on content-based analysis. The feature representation, extraction and matching method would, if possible, reflect human perception. If the

system is reliable enough, it may be able to provide annotations for tagging 3D articulated geometry models automatically. Such annotation may then be used as supplementary information for other retrieval related system.

3. **Reliability** The method should be able to handle similar and dissimilar skeleton models. Similar skeleton models refer to articulated models that are similar in skeletons but different in shapes (e.g., dogs and wolves). Dissimilar skeleton models refer to articulated models that are dissimilar in skeletons (e.g., boys and dogs). We would like to develop a method to handle both of these models. The retrieval performance should be as good as existing works.

4. **High Efficiency** The method should support retrieval in a way faster than sequential search and be able to handle large databases.

3.2 Challenges

After setting out the above research goals, we try to analyze the challenges here. As discussed in our literature review, there are three approaches to handle 3D articulated geometry models. The *Pose-normalizing* approach allows the use of general 3D matching methods. The *Single feature vector* approach produces compact representation and the matching methods are fast. However, these methods cannot handle highly similar skeleton models because fine comparison is not available. To better handle similar skeleton models, the *Bag of features* approach would be the choice. It allows matching in detail. The similarity measure better approaches human perception also. However, most of these works present results that concern dissimilar skeleton models only. They are presented as proofs of concept. As we have observed, there is no large database targeting at highly similar skeleton models so far. Another important challenge is that they are all slow. Most of them are based on non-metric distance where metric indexing schemes are not applicable. The features of both *Single feature vector* and *Bag of features* approaches are all high dimensional. This suggests that they will unavoidably suffer from the curse of dimensionality.

3.3 Proposed Solutions

We propose several solutions to achieve our research goals. We focus ourselves on two areas: reliability and efficiency. We also propose an application for 3D motion retrieval by applying a bag-based matching technique.

3.3.1 Reliability

As pointed out in the literature, the *bag of features* approach should give a better feature matching algorithm. In our earlier work [15] [19], we have developed a bag-based matching method. The feature extraction is based on Level Set Diagram. However, such feature extraction method, as pointed out in relevant works, suffers from stability problems. In the first part of the research, we try to define a better feature extraction algorithm to solve these issues. We expect that with more reliable feature representation, it can lead to more accurate matching results. We then use Earth Mover Distance [86] to define the similarity measure.

3.3.2 Efficiency

We have also observed that Earth Mover Distance is a metric distance measure under certain constraints. By exploring this fact, we might be able to define the first metric similarity measure that allows search of both topological and geometric features in one single step. This is a considerable speed-up compared to the two-step indexing approach as pointed out in our earlier discussion (Section 2.3.4).

However, high dimensionality is an important issue that will degrade retrieval performance. This affects both the *Single feature vector* and the *Bag of features* approaches. In the second part of the research, we propose to use dimension reduction techniques to reduce the intrinsic dimensions of our features. Reducing dimension suggests that we can avoid the problem of dimensionality.

We further propose to use the approximate embedding approach to index and fast search. Though false dismissal may be introduced, the framework is applicable to both metric and non-metric distance measures. We first embed these pairwise distances into vector space. By using existing spatial indexing techniques, high

efficiency is achieved for searching 3D articulated geometry models.

3.3.3 Applications

3D motion is another important area of graphics application. The motion data is used to drive the animation of 3D articulated geometry models. We also propose to apply our idea to convert temporal structure matching into bag-based matching. The idea is to break up motions into different segments and extract features to represent these segments. By using Earth Mover Distance, a metric similarity measure might also be defined if certain constraints are met.

Chapter 4

Feature Extraction and Matching for 3D Articulated Geometry Models

4.1 Introduction

In our previous research [19], we developed a feature extraction and matching method to handle 3D articulated geometry models. We introduced a *critical point analysis* to identify critical points and a *Bi-directional LSD method* to capture bounded regions as features. However, the method is not stable to work on arbitrary meshes. When a model has complicated structure, the method fails. The similarity measure is also non-metric because the computed bounded regions are allowed to overlap with neighboring regions. Since it is non-metric, no indexing technique is applicable. This suggests that it does not scale well to large databases. In this chapter, we try to tackle all these issues. We summarize our contributions here:

1. We present a reliable feature extraction algorithm based on the idea of our previous work [19]. Instead of using bounded regions, we propose to use DMSA to capture reliable topological points and rings. The method, named "Topological Point Ring Analysis", solves the stability problem of our previous work. As

shown in our experiments, the new method produces more accurate matching results than our previous work [19]. It also outperforms Multiresolution Reeb Graph [17].

2. We also focus on designing a *metric* similarity measure which is based on Earth Mover Distance. Special focus is put on ensuring that the metric properties are met. Since the similarity measure is metric, we further implement a distance-based indexing technique, Vantage-Point tree, for fast pruning of irrelevant data. As demonstrated in our experiments, to query for the most similar model, our method only requires 41% of the time to sequentially scan the whole database. This is the first work that can support indexing and search of both topological and geometry features in one single method. It is a considerable speed-up especially when the database contains many highly similar skeleton models.

4.2 Overview of Our Method

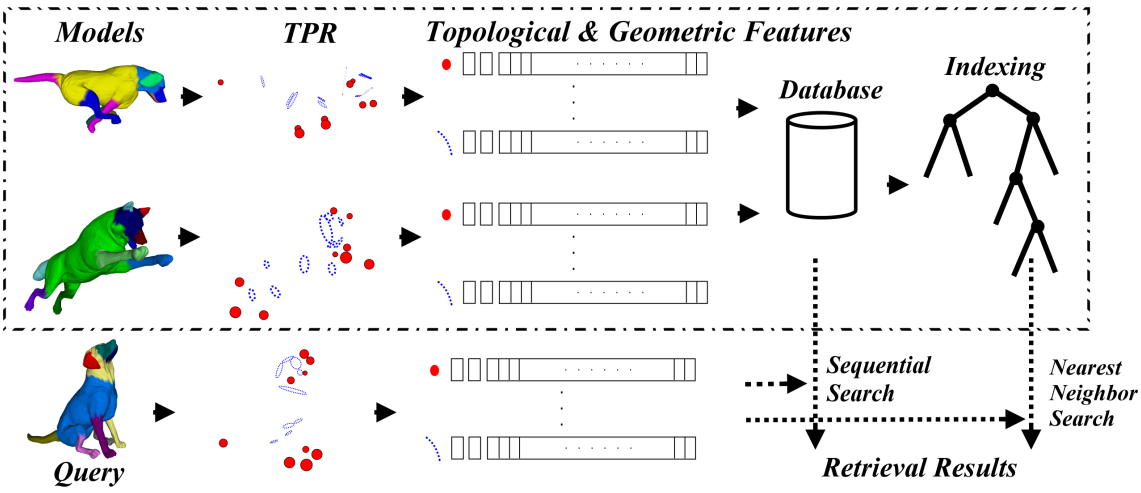


Figure 4.1: Overview of the Retrieval System. A *Topological Point Ring (TPR) analysis* is proposed to capture topological and geometric features for matching. We also define a metric measure that allows indexing for fast search.

The focus of our method is to develop a feature extraction and matching algorithm to handle 3D articulated geometric models. We extract two types of topologi-

cal features: topological points and topological rings, and several geometric features. Since the following discussion involves several steps, we try to give an overview of our method in Figure 4.1.

Topological Features: Intuitively, a topological point refers to a salient point located at a protrusion tip, and a topological ring refers to a border that separates two significant components in a model.

To capture topological points to represent protrusion tips, we first derive our algorithm from a skeleton extraction technique, the Level Set Diagram (LSD) [40]. However, LSD suffers from two problems: extraneous critical points [87] and slicing direction [88]. The former refers to the fact that many redundant critical points are extracted even on smooth surface. The later means that the location of critical points are affected by a certain slicing direction. We will discuss these two problems more in Sections 4.3.4 and 4.4.

To alleviate these problems while remaining fast and automatic, we propose a method, *Topological Point Selection*, which is described in detail in Section 4.5.1. The method produces validated maximum critical points, referred to as topological points. To reduce computation time, we further discuss how we select the minimum critical points (source points) in Section 4.5.2. Our method also uses topological rings as features. To extract reliable topological rings to represent joint locations, we propose *Topological Ring Extraction* in Section 4.5.3. We name the whole feature extraction method as *Topological Point Ring (TPR) analysis*.

It should be noted that Mortara et al. first proposed the term "Topological Ring" in [88]. The method uses topological expansion to define such rings. However, the method assumes regular mesh sampling which may not be useful for general triangulated mesh. Being inspired from [88], we develop our own extraction method based on geodesic distance and use the term "Topological Ring" because the basic extraction idea is similar. In our method, we use critical points from Morse Theory (i.e. Level Set Diagram), which studies the topology of smooth surface, to define our feature points. Therefore, we refer to these feature points as "Topological Points" in our context.

Geometric Features: After obtaining all topological features (points and rings), we extract geometric features to characterize each of them. There are two kinds of geometric features in our method: local and global features. Local features include normalized integral geodesic and effective area. They are used to characterize the locations and importance of a topological feature. Global geometric features are used to capture the surface information of a model. They help discriminate similar skeleton models like girls and babies. Hence, our model signature is defined by a collection of topological features and each of them is characterized by a number of geometric features. It should be noted that these features were first proposed in our earlier work [19]. Since this feature representation is essential to the derivation of a metric similarity measure, we describe it briefly in Section 4.5.4.

Matching and Indexing: We formulate the matching of two models as energy transfer between two signatures by adapting the EMD, which computes the minimum energy required to transform one signature into another. In this work, particular focus is put on ensuring that the feature representation and ground distance follow triangle inequality. We define our metric distance function for the EMD matching framework in Section 4.6. Since the function is a metric, we can construct a fast indexing scheme by building a VP-tree. Such an indexing scheme can support searching of both topological and geometric features in one pass.

4.3 Background Knowledge and Our Earlier Works

4.3.1 Notation

We provide a brief notations for our coming discussion.

- DSSA - Dijkstra's Single Source shortest path Algorithm.
- DMSA - Dijkstra's Multiple Source shortest path Algorithm.
- $g_q(p)$ - Geodesic distance measured from q to p.
- $G(q)$ - Integral Geodesic.
- l_s - A level is a contour defined on the surface such that all points of the

contour have the same scalar value. In this work, we always assume the scalar value of a point corresponds to the geodesic distance between the point and a source point s.

- $C(l_s)$ - A level set, which is the realisation (polygonal contour) of a level l.
- Max_s,$Saddle_s$,Min_s - The sets of maximum, saddle and minimum critical points, respectively. These critical points are defined by analysing the scalar values around a point, according to the Level Set Diagram method.
- U - A topological feature, which would be a Topological Point or Ring.
- S - a triangulated mesh S is a set of vertices $\{v_i\}$, edges $\{v_i, v_j\}$ and faces $\{v_i, v_j, v_k\}$.

4.3.2 Integral Geodesic

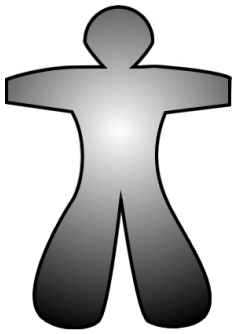


Figure 4.2: Integral geodesic on a surface. The brighter region is closer to the surface center, whereas the darker regions are farther away from the surface center.

Geodesic and integral geodesic are basic to our method. Hilaga et al. [17] first suggest the use of integral geodesic, which is defined on an arbitrary surface mesh as $G(q) = \int_{p \in S} g_q(p) \partial S$. Given a vertex q, integral geodesic is the integral of all geodesics g measured from q to all vertices p on a surface S. Geodesic $g_q(p)$ is the shortest distance between two points g and p on a surface. In our work, we use Dijsktra's Single Source shortest path Algorithm (DSSA) to approximate geodesic.

In general, integral geodesic gives a small scalar value if vertex q is near to the center of the mesh (brighter region in Figure 4.2) and a larger scalar value if q is located away from the center (darker region in Figure 4.2). Hence, integral geodesic

indicates how far a vertex is from the points that have minimum integral geodesic. Note that a point with minimum integral geodesic is not the *center of mass* of the model; the *center of mass* can be considered as the minimum integral Euclidean distance of a point set. With the analog of center of mass, we refer to the points having minimum integral geodesic as the *surface centers* as it is defined on the surface. Figure 4.2 shows the function of integral geodesic on a surface.

4.3.3 Level Set Diagram (LSD)

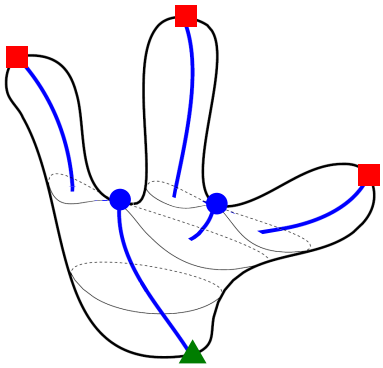


Figure 4.3: Level Set Diagram and Morse Theory. Maxima: Square at finger tips, Saddles: Dots at branches, Minima: Triangle at wrist (the origin of the graph).

The LSD [40], which is based on the Morse theory, is a skeleton extraction technique. The Morse theory describes how the differential geometry of a surface algebraically relates to the topology. LSD applies the theory on polyhedral surfaces to extract skeletons. It uses geodesic distance $g_s(v)$ as the Morse function to define a scalar value for each vertex on the surface.

In order to define topological change at a vertex v, a $index(v)$ function is defined for each vertex. Let $w_1, w_2, ..., w_k$ be the k neighbors of v enumerated counterclockwise around v, the number of sign changes in the sequence $(g_s(w_1) - g_s(v), g_s(w_2) - g_s(v), ..., g_s(w_k) - g_s(v), g_s(w_1) - g_s(v))$ is defined as $Sgc(v)$. The $index(v)$ is defined as:

$$index(v) = 1 - \frac{Sgc(v)}{2} \tag{4.1}$$

Such $index(v)$ is then used to extract three kinds of critical points (minima, saddles, and maxima) as shown in Figure 4.3. Formally, maximum and minimum critical

points have $index(v) = 1$, where minimum critical point is the source point of the geodesic distance, s. The saddle critical points have $index(v) < 0$, whilst an ordinary vertex has $index(v) = 0$. (Figure 4.4)

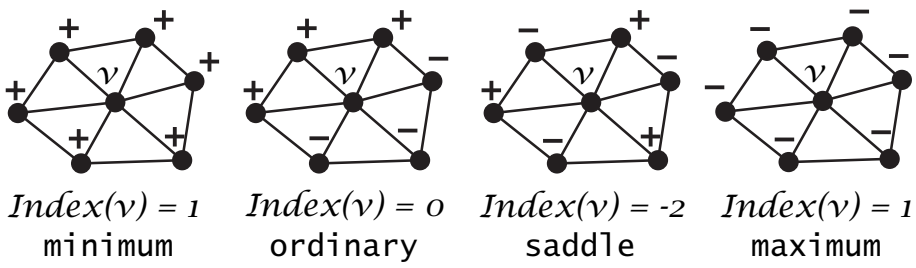


Figure 4.4: Definition of Ordinary and Critical Points

The LSD skeleton is defined by connecting every average point of adjacent levels. LSD defines such level by tracing the geodesic wavefront through a level set $C(l_s)$ (Figure 4.5), which is a polygonal contour of the same level l_s on the surface, where l_s is a scalar value defined on the surface with respect to a source vertex s. An edge (v_i, v_j) is called a cross-edge if it passes through level l_s satisfying the condition: $g_s(v_i) < l_s < g_s(v_j)$, where $g_s(v)$ is the scalar value at vertex v obtained by calculating the geodesic from a minimum (source) point s. LSD uses the DSSA to approximate the geodesic distance.

In our context, instead of tracing every level and construct a skeleton, we simply compute these critical points and build an LSD tree with the minimum, saddle, and maximum points forming the root, the internal nodes, and the leaf nodes of the tree, respectively.

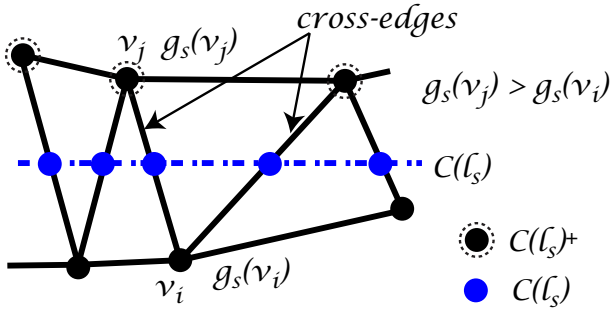


Figure 4.5: Realization of a Level Set and Cross-Edges

4.3.4 Critical Point Analysis

As mentioned, our topological features are composed of topological points and rings. We choose the maximum critical points from LSD as the topological points because the locations of maximum critical points match the idea of topological points discussed in Section 4.2. Though it is easy to apply LSD, there are two problems that hinder us from using it directly here: extraneous critical points and slicing direction. We discuss extraneous critical points in this section and slicing direction in Section 4.4.

Shortest path algorithms usually suffer from getting extraneous critical points when they are applied on meshes. As LSD is based on computing shortest path distances (geodesics), it also suffers from this problem. According to [88], extraneous critical points may result from noise, precision errors, or the fact that geodesic distance is not a good Morse function. Though [88] provides a method to find the optimal number of critical points, it is not fully automatic, making it less suitable for use in a 3D model search engine. To alleviate this problem, we use a modified LSD method, *Critical Point Analysis*, that we have proposed earlier in [46].

It is observed that extraneous points arise during the registration of saddles, and these critical points are very close to each other. As such, applying a proximity-filtering step before registering a saddle would help alleviate such problem.

Before a vertex v is registered as a saddle in LSD, we approximate the level set $C(l_s)$ by defining a vertex set $C(l_s)^+ = \cup v_j$ such that $g_s(v_j) > l_s > g_s(v_i)$ for all cross-edges (v_i, v_j). The relation of $C(l_s)$ and $C(l_s)^+$ is illustrated in Figure 4.5. Such $C(l_s)^+$ can always be split into disjoint cycle vertex sets $CycleVS_i$ because a level set of a Morse function, which is defined on a closed smooth surface, is composed of disjoint closed curves, if that level set contains no critical points [40]. In Figure 4.6, we illustrate a saddle vertex with 3 cycle vertex sets $CycleVS_{1,2,3}$. We have also shown $C(l_s + \epsilon)$ where ϵ is a small value. We show $C(l_s + \epsilon)$ instead of $C(l_s)$ because it does not contain the saddle critical points and can always be split into three disjoint closed cycles. The reason to compute $C(l_s)^+$ is that we can quickly obtain all cycle vertex sets by a depth-first search on the running heap of DSSA.

In the filtering step, a vertex v is considered a valid saddle if and only if the

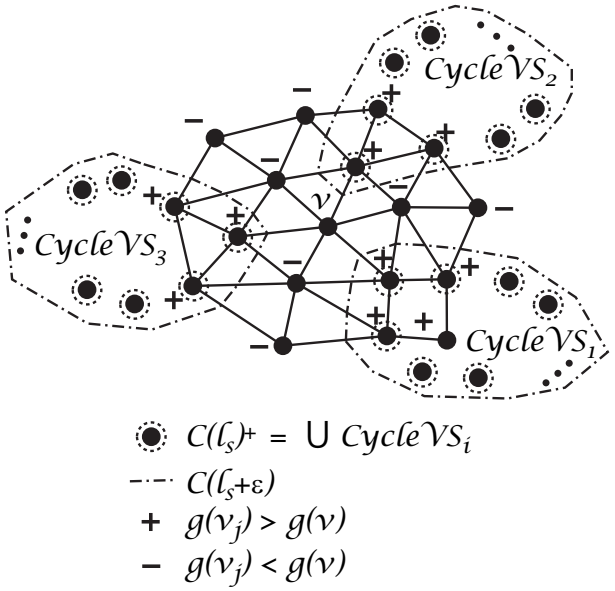


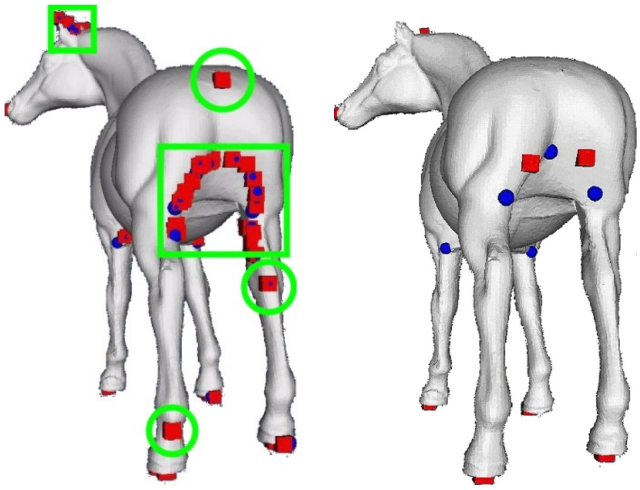
Figure 4.6: Example of $C(l_s)^+$ and its three cycle vertex set

number of vertices inside all $CycleVS_i$ is greater than a number n, where n indicates how strong the filtering step is. If the number is small, it allows a smaller distance between adjacent critical points. Generally, we choose $n = 1$ so that it will not miss small features. Figure 4.7 shows an example before and after applying the *Critical Point Analysis*.

4.3.5 Review of Our Earlier Work

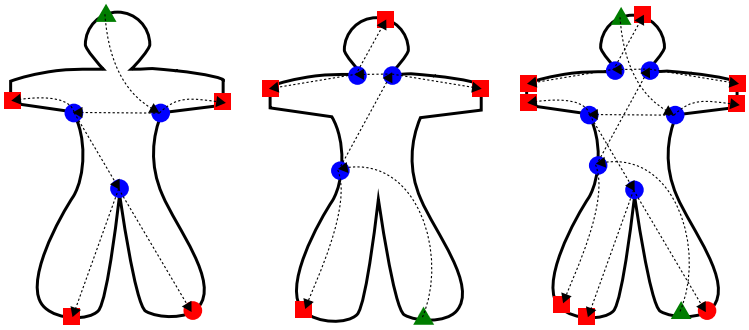
In our earlier work, we proposed a feature extraction algorithm called Bi-directional LSD (BDLA). It proceeds as follows. We first apply LSD on two vertices that are furthest apart (Figures 4.8(a) and 4.8(b)). Then we obtain two LSD trees and so two sets of maximum, minimum and saddle critical points (Figure 4.8(c)).

Since the two LSD trees are very similar, we may pair up these saddle and maximum critical points, and so extract various bounded regions using simple geodesic region growing. For example, to extract a bounded region on an arm, we first compute a middle vertex z between two maximum critical points (from two different trees). Then, we analyse the two LSD trees and look for pair of two saddles points in the trees, where these saddles are parents of the paired maximum critical points. We further compute the perpendicular distance to the line joining the two



(a) Before (b) After

Figure 4.7: Examples before and after applying *Critical Point Analysis*. Extraneous critical points are enclosed in green.



(a) 1st LSD (b) 2nd LSD (c) BDLA

Figure 4.8: Review of Bi-Directional Level Set Diagram. Blue circle: saddle points; Red dot: maximum point; Yellow triangle: source point.

saddles. This perpendicular distance gP defines the radius of the bounded region $= \{v \in S | g_z(v) <= gP\}$ (see Figure 4.9). The method then continues using the remaining saddle critical point pairs to define other bounded regions.

4.4 Problems and Proposed Solutions

After briefly reviewing BDLA, we observe two problems in the method. As mentioned in Section 4.3.4, feature extraction based on solely critical points may lead to unreliable features. There are two main problems: extraneous critical points and

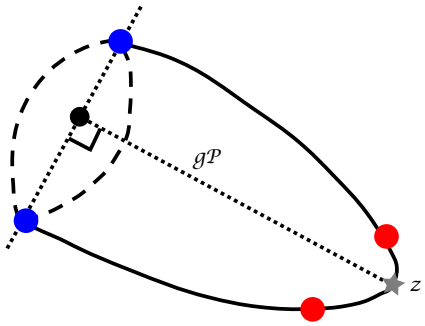


Figure 4.9: Bounded Region Extraction of the BDLA method.

slicing direction. BDLA is based on our *Critical Point Analysis* only, and so only the first problem is alleviated. The second problem is not solved.

Slicing direction refers to the situation that the extraction of critical points may be favored by a particular slicing direction. In short, the algorithm can yield a totally different critical point set when a different source point is chosen [88]. A simple example can be seen in Figure 4.8. In the figure, all saddles critcal points in the two LSD trees are different. This suggests that the use of saddle critical points are unreliable.

Another shortcoming of the method is that it assumes to have two nicely structured LSD trees where it is always able to find pair of saddle points from the two LSD trees to define bounded regions. However, when LSD is applied on general mesh, it would create totally different LSD trees, and so matching pair of saddle points become difficult. One example can be seen in Figure 4.7. We can see that, though after filtering, there are still some maximum and saddle points at the rear of the horse which do not correspond to any protrusion. If we apply LSD from another source point, for example, from one of the rear legs, these maximum and saddle points will not be extracted. This demonstrates that it would create a pairing problem and so, practically, BDLA is not robust on general model where the LSD trees cannot provide a proper pairing of saddle points. Finally, the bounded regions extraction are based on simple geodesic inequality. For example, a finger (a protrusion region) is defined as $= \{v \in S | g_z(v) <= gP\}$ where z is the middle vertex discussed earlier. However such definition does not exclude the overlapping region from a nearby finger (see Figure 4.10 for an illustration). This violates the metric

constraints of the Earth Mover Distance, and so the resulting similarity measure is not metric.

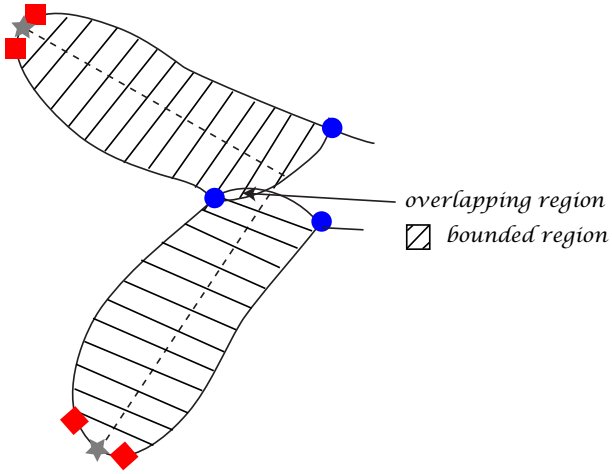


Figure 4.10: Illustration of Overlapping Bounded Region of BDLA method

To solve these problems, we first consider topological points and rings instead of bounded regions as features. Second, we design an algorithm to extract these features using Dijkstra's Multi-Source shortest path Algorithm (DMSA). This solves the slicing direction problem. We also make sure the weights (importances) associated with these features meet the metric constraint of the Earth Mover Distance. The whole scheme can then be indexed for fast search. We discuss all these in the following sections.

4.5 Topological Point Ring (TPR) Analysis

In this section, we discuss our new feature extraction algorithm, *Topological Point Selection* and *Topological Ring Extraction*.

4.5.1 Topological Point Selection

Apart from extraneous critical points, LSD also suffers from the slicing direction problem. To tackle this stability problem, we introduce the idea of using multiple source points and derive *Topological Point Selection* as a process for selecting valid topological points. Suppose that we have n source points, $s_1 ... s_n$, on the surface.

We apply *Critical Point Analysis* on each of these n source points to obtain 3 sets $Max = \cup_{i\in(1..n)} Max_{s_i}$, $Min = \cup_{i\in(1..n)} Min_{s_i}$ and $Saddle = \cup_{i\in(1..n)} Saddle_{s_i}$, where Max_{s_i} and $Saddle_{s_i}$ are the sets of maximum and saddle critical points with respect to a source point s_i. The minimum critical point set $Min_{s_i} = \{s_i\}$ contains the source point s_i. Since the location of maximum and minimum critical points corresponds well to our idea of topological point, we take them as the potential topological point set, and call them extreme points $Extreme = Max \cup Min$.

Our *Topological Point Selection* then identifies valid topological points by counting the number of different extreme points nearby. Let the search region be:

$$SearchRegion(m) = \{q \in S | g_m(q) \leq g_m(sd(m))\},\ m \in Extreme \tag{4.2}$$

The search radius is set to be the geodesic distance measuring from a point m to the nearest saddle critical point $sd(m) = \arg\min_p g_m(p), p \in Saddle$. Let the set of extreme points in the search region be

$$\xi(m) = \{v \in Extreme \cap SearchRegion(m)\} \tag{4.3}$$

and the set of different LSD trees that produce extreme points in the search region be:

$$ExtremeTree(m) = \{i \in (1..n) | v \in \xi(m) \text{ and } v \in Max_{s_i} \cup Min_{s_i}\} \tag{4.4}$$

In our method, if $\|ExtremeTree(m)\|$ in the search region is more than $\frac{n}{2}$, that is, more than half of the LSD trees that produce extreme points in the search region (a majority vote), we consider m as a valid topological point. We repeat this for all $m \in Extreme$. Since there may be many valid topological points in a region, we choose the one that possesses the furthest distance from its sd as the topological point in that region. Therefore, the definition of Topological Point is:

$$\{m \in Extreme | \|ExtremeTree(m)\| > \frac{n}{2}, m = \arg\max_v g_v(sd(v))\} \tag{4.5}$$

where $v \in \xi(m)$.

4.5.2 Source Point Selection

To perform Topological Point Selection, a number of source points must be selected. We have found from experiments that three source points, if selected appropriately,

are sufficient to identify all valid topological points with minimal computational cost. We choose the two furthest points in a 3D mesh as the source points. The first point can be found by running Dijkstra's single source shortest path algorithm (DSSA) on an arbitrary point on the model to obtain the point with maximum geodesic. The second point can then be determined by applying DSSA again on the first point to obtain another point with maximum geodesic [40]. However, since they are located at the far end of a model, LSD may still favor a particular direction and miss some critical points. Thus, we choose the third source point near the center of a mesh. A good choice is the surface center that we have discussed earlier in Section 4.3.2.

In [17], an approximation method is proposed to find the surface center by sampling at least 120 points on the surface. We observe that most articulated models are not perfectly symmetric and, usually, there is only one point that corresponds to the minimum integral geodesic. To speed up the process, we apply a hierarchical search to locate the point. (Note that our method still works even if there is more than one such point in a mesh as we only need to find a reference point.) In our hierarchical search, we first split the surface into many patches. For each patch, we calculate the integral geodesic at the patch center. We then identify the patch with the smallest integral geodesic. We split it again into many subpatches and calculate the integral geodesic at each subpatch center. We apply this strategy recursively until the change in the smallest integral geodesic is less than a threshold or the patch area is too small. Figure 4.11 shows the topological points obtained from Topological Point Selection and the LSD tree constructed using Critical Point Analysis with the surface center s being the source point.

4.5.3 Topological Ring Extraction

The term "Topological Ring" is first mentioned in [88]. Given some initial points, the method applies topological expansion of a 1-ring neighborhood. When frontiers of different topological expansions collide, a branching is identified. Mortara and Patane [88] define the borders of these frontiers as topological rings. However, as their objective is on skeleton extraction, the topological rings produced are not

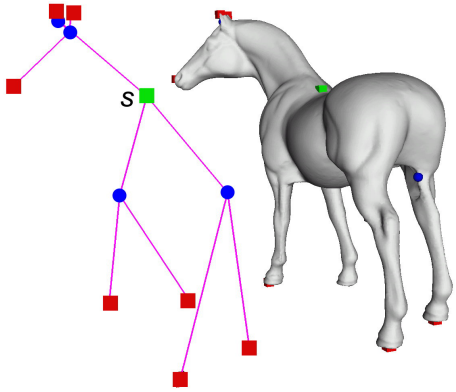


Figure 4.11: *Topological Point Extraction*. Square boxes (except the source point) are topological points. The LSD tree is constructed by *Critical Point Analysis*.

suitable to be used as features here for two reasons. First, the topological expansion in [88] depends on a 1-ring neighborhood only. To extract topological rings reliably for use as features, a regular tessellation of the mesh surface is required. Second, the method processes all source points at the same time and, thus, the locations of topological rings greatly depend on the differences in branch lengths. For general 3D models, these two requirements may be difficult to satisfy. Hence, the method cannot ensure consistent recovery of topological rings.

Here, we propose our Topological Ring Extraction to address this problem. Our method is similar to [88] in that it also features a multi-source point approach. However, instead of using topological expansion of 1-ring neighborhood, we use shortest path growing, and so it does not require regular tessellation of the mesh surface. In addition, different source points are given different initial values. This allows more stable extraction of topological rings with no regard for branch lengths. Here, we give a definition of our proposed Topological Ring.

4.5.3.1 Topological Rings

First, let us consider a scalar function defined on the surface.

$$F(v) = \max_{q \in S} g_o(q) \times \frac{\max_{q \in S} G(q) - G(v)}{\max_{q \in S} G(q) - \min_{q \in S} G(q)} \tag{4.6}$$

Intuitively, $F(v)$ (Figure 4.12) is the same as Integral Geodesic $G(v)$ but normalized to $(0, \max_{q \in S} g_o(q))$, having the largest value at the surface center(s), o, and smallest

value at the point furthest away from it. It approximates the geodesic distance from a point to its closest surface centers. This scalar function on the surface provides an initial value for every topological point (Section 4.5.1) to be used in our multi-source methods.

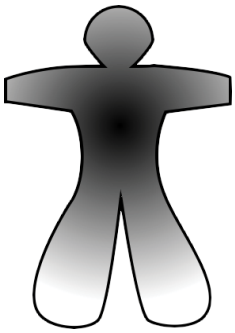


Figure 4.12: The Function on the Surface for Setting Initial Values

To define a neighborhood structure with respect to the geodesic distance from a vertex set Ω, we can construct $N(\Omega)$ as follows:

$$\begin{aligned} N(i) &= \{j \in S | g_i(j) \leq F(j) - F(i)\} \\ N(i,k) &= \{j \in S | g_i(j) \leq F(j) - F(i) \vee g_k(j) \leq F(j) - F(k)\} \\ \text{Similarly,} N(\Omega) &= \left\{ j \in S \;\middle|\; \begin{array}{cc} g_i(j) \leq F(j) - F(i) & \vee \\ \ldots & \vee \\ g_k(j) \leq F(j) - F(k) & \end{array} \right\}, \Omega = \{i, ..., k\} \end{aligned} \tag{4.7}$$

This states that $N(i)$ of a vertex i is the set of vertices q on the surface such that its geodesic distance from i is less than the difference of the value $F(j) - F(i)$. When we are considering the neighorhood of more than one vertex e.g. (i,k), it is defined as a simple union operation ("or" condition). And finally, the neighborhood structure of the vertex set Ω is defined by generalizing the above definition.

We can further define a local mesh system, including the edges and trianges as follows:

$$T(N(\Omega)) = \bigcup_{l,p,q \in N(\Omega) \wedge l,p,q \in S} T(l,p,q) \tag{4.8}$$

where $T(l,p,q)$ is the triangle with vertices l, p, q.

Let $\bar{\Omega}$ be the set of all topological points. We refer to the largest growing region

without colliding with others as:

$$Region(N(\Omega)) = \arg \max_{T(N(\Omega))} |N(\Omega)| \text{ s.t. } i \in N(\Omega) \wedge i \notin N(\bar{\Omega} - \Omega), i \in S \quad (4.9)$$

where $|N(\Omega)|$ is the number of vertices in the neighborhood structure. This states that a mesh region is a set of triangles (include vertices and edges) $T(N(\Omega))$ where $|N(\Omega)|$ is maximized. Also, every vertex i in $N(\Omega)$ would only be geodesically closer to the set of Ω, but not the rest of possible $(\bar{\Omega} - \Omega)$. If one vertex, j, is closer to other topological points $(\bar{\Omega} - \Omega)$, a new neighborhood structure has to be constructed which suggests a merge, i.e. $N(\Omega \cup \Omega')$, where $\Omega' \subset \bar{\Omega} - \Omega$ is the set of topological points of which $j \in N(\Omega \cup \Omega')$ is closer to. In the multi-source growing algorithm, such situation is detected as a collision of two growing frontiers.

Therefore, we can provide the definition of a Topological Ring as the vertex set of the border of a maximum growing region without collision:

$$\{v \in \partial Region(N(\Omega))\} \quad (4.10)$$

where ∂ is the border operator.

To reduce the number of redundant topological rings, we can further consider the *Length* of a structure.

$$Length(N(\Omega)) = \max_{i \in N(\Omega)} F(i) - \max \left(\max_{j \in N(\Omega_1)} F(j), \max_{k \in N(\Omega_2)} F(k) \right) \quad (4.11)$$

where $\Omega = \Omega_1 \cup \Omega_2$ and this implies a valid merge. We only take $\partial Region(N(\Omega))$ as a valid topological ring, if $Length(N(\Omega)) > \frac{1}{2} \max(Length(N(\Omega_1)), Length(N(\Omega_2)))$.

After defining the above, we observed that though the scalar function $F(m)$ provides a guidance for region growing, it requires to exhaustively compute integral geodesic for large amount of vertices. Therefore, in the following section, we discuss some practical implementation issues. The method includes three parts: initialization, shortest path algorithm, and termination. The initialization stage defines different initial values for different topological points. Our shortest path algorithm then traces the geodesic wavefronts from these topological points using the corresponding initial values. When executing the algorithm, different wavefronts merge and new wavefronts are formed, which become the topological rings. When all frontiers merge into one, the algorithm terminates. The details of these three stages are discussed as follows.

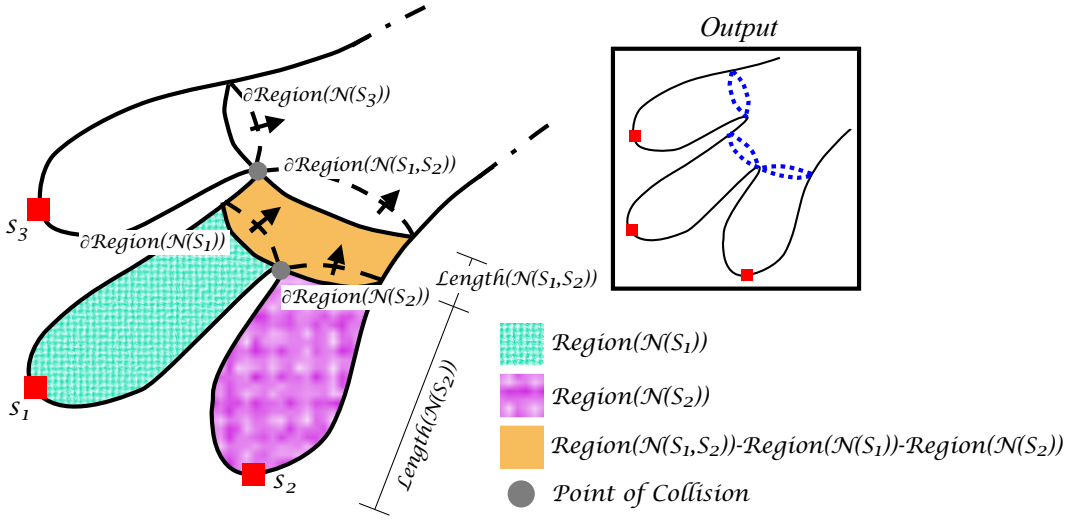


Figure 4.13: *Topological Ring Extraction.* Shortest path growing and validation with three topological points.

4.5.3.2 Initialization

In this stage, we determine an initial value for each topological point such that topological rings may collide near joint locations. We assign a smaller initial value to a topological point that is far away from the mesh center and a larger value to one that is near the mesh center. This initial value is the starting time of the shortest path algorithm for that point. Since all topological points are located far away at protrusion tips, these initial values ensure that the shortest path algorithm is always moving toward the center. We use the *surface center* to approximate the mesh center. Because integral geodesic is a good measure of the relative distance from the *surface center*, o, we compute the initial value of a topological point m based on $F(m)$. Recall that we have obtained an approximation of surface center, o, earlier in Section 4.5.2. Therefore, we have obtained $\min_{q \in S} G(q)$. $\max_{q \in S} G(q)$ can be obtained by computing integral geodesic at every topological point m, and finding the maximum one because topological points are located at protrusion tips, and they are always further away from the surface centers. Given all these, $F(m)$ can be evaluated at topological points.

4.5.3.3 The Shortest Path Algorithm

After defining the initial values for the topological points, we put these points in the heap of a DMSA. The topological expansion is, in effect, a shortest path algorithm. During the execution, the vertex with the smallest initial value is removed from the heap one at a time and its neighbors are updated. Hence, we may consider that different geodesic wavefronts grow from different topological points and move toward the *surface center*. It should be noted that during algorithm execution, we do not check the $F(i) - F(j)$ conditions because DMSA is approximating such value by geodesic distance directly. Also, DMSA always runs toward the center and will not visit vertices that have been visited. This also makes sure that the condition is kept all the time. Though there are slight differences between the two, we find that it is sufficient to use DMSA directly. When executing the algorithm, if two geodesic wavefronts meet, a merge of geodesic wavefronts occurs and a new wavefront is formed. This is represented as $N(\Omega_1 \cup \Omega_2) = N(\Omega)$, where $N(\Omega)$ is the new growing structure, and $\partial Region(N(\Omega_1))$ and $\partial Region(N(\Omega_2))$ become two topological rings.

We mentioned earlier that topological rings are registered when geodesic wavefronts meet. However, redundant topological rings may sometimes be created. For example, in Figure 4.13, we would expect to have three rings located at the three joints between the palm and the fingers as shown inside the square box. To detect this case, we check if a geodesic wavefront is a valid topological ring by measuring the *Length* (Eqn 4.11) of the region formed as defined earlier.

4.5.3.4 Termination of the Algorithm

The algorithm terminates when all geodesic wavefronts have merged into one. However, we note that, when we have visited all saddle points obtained from Critical Point Analysis using the *surface center* as the source point, there should be no more branching left and the algorithm may end. The unvisited vertices or vertices that are visited but not yet included in any region are grouped into one final region (FR). Figure 4.14 shows some example models with various topological rings extracted using our algorithm.

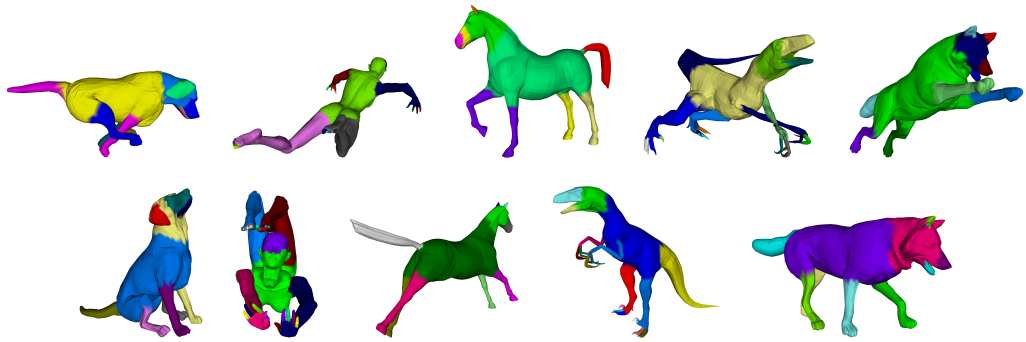


Figure 4.14: Topological Rings (borders between different colored regions).

4.5.4 Geometric Feature Extraction

After TPR analysis, we obtain a set of topological points and rings together with a set of regions. These topological points and rings are located at protrusion tips and articulated joints. They provide skeletal information of the model, independent of model articulation. We characterize each of these topological features with three types of geometric information: Normalized Integral Geodesic, Effective Area, and Geometric Surface Vector.

4.5.4.1 Normalized Integral Geodesic - Spatial Information

Integral geodesic is a function defined over the surface to indicate how far a point is from the *surface center*. It maps a point to a scalar value and is thus a good feature to describe the spatial location of a topological point. To generalize this function to any topological feature (point or ring), we need to consider the case of the topological ring as well. To compute the integral geodesic for a ring, we interpolate the value from the integral geodesic of the *surface center* and the value of one of the originating topological points of the ring. Since a ring may come from many originating topological points, we use the one that is furthest away from the ring, which is the ancestor topological point as its distance from each vertex on the ring has a smaller deviation. The interpolation requires two distance values: from the ring to the *surface center* and to the ancestor topological point. To compute these distances, we use geodesics with respect to a vertex set (vs). Such distance, which is denoted as g_{vs}, can be calculated by Dijsktra with all the vertices in the

ring as source points. The generalized Integral Geodesic $G'(U)$ of topological feature (point or ring) U is computed as follows:

$$G'(U) = \left\{ \begin{array}{cc} G(U) & \text{if U is a topological point} \\ \frac{g_{vs}(o,U)\times G(w)+g_{vs}(w,U)\times G(o)}{g_{vs}(o,U+g_{vs}(w,U))} & \text{if U is a topological ring} \end{array} \right\} \tag{4.12}$$

where $G(v)$ the integral geodesic of point v. $g_{vs}(o,U)$ and $g_{vs}(w,U)$ are the geodesic distances measured from topological ring U to *surface center* o and to ancestor topological point w, respectively. Finally, we calculate the Normalized Integral Geodesic as follows:

$$G'_{norm}(U) = \frac{G'(U) - \min_{\forall q \in S} G(q)}{\max_{\forall q \in S} G(q) - \min_{\forall q \in S} G(q)} \tag{4.13}$$

4.5.4.2 Effective Area - Weights of Importance

We note that the importance of a topological ring located in a finger, for example, should be smaller than that located in the leg. This is intuitive as removing a leg from a 3D model gives a larger perceptual impact than removing a finger. Hence, we approximate the importance of topological features by distributing the local surface areas among the adjacent topological features. We denote such a redistributed area as the Effective Area. To simplify our discussion, we first define some abbreviations. A Protrusion Region (**PR**) is a region bounded by a topological point and a topological ring. A Segment Region (**SR**) is a region bounded by topological rings only. An **FR**, as mentioned in Section 4.5.3.4, is the final extracted (core) region. We consider two cases in our method:

1. **PR** - Simply divide the **PR** surface area into two and associate half to the topological point and half to the topological ring.

2. **SR** and **FR** - Distribute the local surface area to the adjacent topological rings in proportion to $Region(C(l_s)^-)$ computed for each adjacent ring $C(l_s)^-$.

Note that:

$$\sum EffectArea = \frac{\sum Area(\mathbf{PR}) + \sum Area(\mathbf{SR}) + \sum Area(\mathbf{FR})}{Area(S)} = 1 \tag{4.14}$$

The distribution of these effective areas (Eqn 4.14) ensures that the total importance of all topological features (points or rings) equals to 1. This is essential to define a metric similarity measure in Section 4.6.3.

4.5.5 Geometric Surface Vector - Surface Distribution

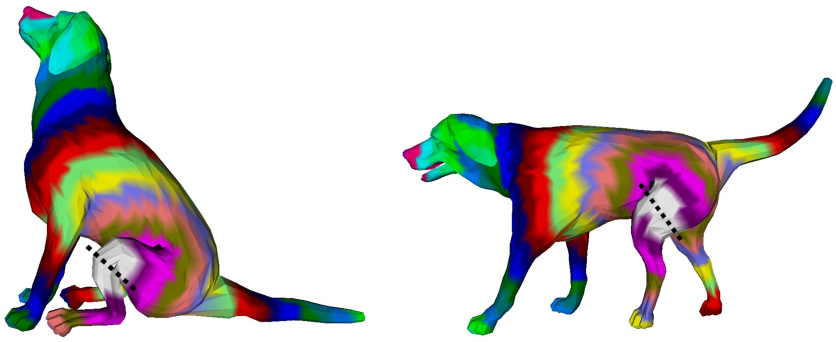


Figure 4.15: Models divided into 20 bands from topological rings (dashed lines).

In order to better discriminate similar-skeleton models, we consider additional geometric information to describe the global surface change. We extract three global geometric feature vectors for each topological feature U (point or ring): curvature $K(U)$, area $A(U)$, and average distance $H(U)$. We construct these three vectors by first dividing the model into 20 bands according to the geodesic distances from a given topological feature U.

$$\text{band}_i(U) = \{v_j \in S | \frac{i-1}{20} \max_{q \in S} g_{vs}(q, U) < g_{vs}(v_j, U) \leq \frac{i}{20} \max_{q \in S} g_{vs}(q, U)\} \quad (4.15)$$

where $i \in 1...20$, and aggregate basic geometric properties of each band into an entry in the feature vectors, i.e. $K_i(U)$, $A_i(U)$ and $H_i(U)$ respecticely. Since we use geodesic distance, the resulting feature vector is stable toward mesh articulation. As an example, we divide two dog models shown in Figure 4.15 into 20 geodesic bands relative to a topological ring located at one of the legs. Bands of the same color indicate that they are within the same geodesic interval from the ring. We can see that although the two dogs have different poses, the locations of the color bands are similar.

To compute these vector entries $K_i(U)$, $A_i(U)$ and $H_i(U)$, we first compute basic geometric properties for each vertex in a band. We follow [89] to compute Gaussian

curvauture for each vertex, where the implementation is provided by [90]. The area of a vertex is $\frac{1}{3}$ the sum of areas of all the triangles around it, because triangle is shared by three vertices. "Average Distance" of a vertex is the distance between a vertex and the center of mass $c = \frac{1}{||\text{band}_i(U)||}\sum_{v\in\text{band}_i(U)} Pos(v)$ of its associated band, where $Pos(v)$ is the operator to return the position of a vertex v. We then compute the $K_i(U)$, $A_i(U)$ and $H_i(U)$ as the average of all these basic geometric values associate to all vertices in the band. In general, area and curvature are used to capture the global surface change, whereas average distance measures the thickness of individual segments. All these features have been discussed in our earlier work [19] and is shown to be stable towards articulation.

Sometimes, a single band may be composed of several segments. For example, some color bands in Figure 4.15 may have segments at different locations like the body, the limbs, and the tail. To improve the descriptiveness of the feature, we apply depth first search to locate all connected components of each band, and use the surface area ratio of these components to prorate the final value of that band.

4.6 Feature Matching

With a signature for each model, that is, a set of topological features (points or rings) each described by three types of geometric features, we may compare the similarity of different models based on matching the signatures. Here, we propose to use Earth Mover Distance (EMD) [86] to define our similarity measure.

4.6.1 The EMD Method

The EMD method is based on the Transportation Problem. Consider representing the features of the query as pieces of mass in space and the features of the candidate as holes in space. The concept behind the EMD method is to calculate the minimum energy required to move the pieces of mass (or earth) to the holes to completely fill the holes.

Transporting a heavy piece of mass a long distance generally requires more energy than transporting it a short distance. Therefore it is beneficial to find a solution to

transport the pieces as short a distance as possible. The energy required to move a piece of mass to a hole is called the *flow*. The total energy to move all pieces of mass is called the *total flow*. In the EMD we try to find an optimal total flow.

More formally, the EMD method computes the minimum energy to move a set of masses, P, to a set of holes, Q, as follows:

$$EMD(P,Q) = \frac{\sum_{i=1}^{m}\sum_{k=1}^{n} d_{ik} f_{ik}}{\sum_{i=1}^{m}\sum_{k=1}^{n} f_{ik}} \tag{4.16}$$

where d_{ik} is the energy required to transport one unit of mass from feature i of the query to feature k of the candidate. f_{ik} is the amount of flow transported from feature i of the query to feature k of the candidate.

To calculate d_{ik}, a Ground Distance metric is required. For each feature i of the query and feature k of the candidate, we compute a Ground Distance value. All the computed values are stored in a *cost matrix*. Figure 4.16 shows an example of this operation. After the cost matrix has been produced, a flow matrix is computed containing the flow from feature i of the query to feature k of the candidate for all i and k. This flow matrix is iteratively optimized to find the optimal total flow.

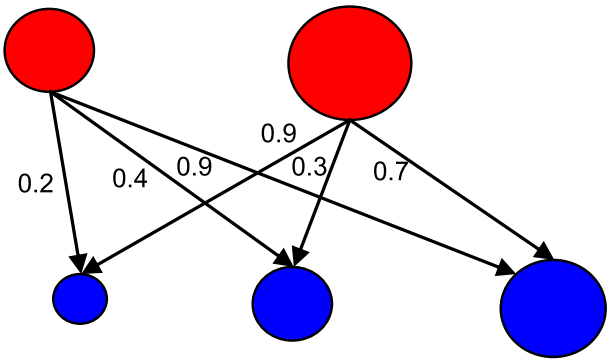


Figure 4.16: Example of *ground distances* between each pair of features that produce the cost matrix.

4.6.2 Similarity Measure

In our approach, we consider a topological feature as an EMD point and define Effective Area as weight. To describe the energy transfer between two EMD points, we further define a distance function $Dist()$ based on geometric features as follows:

$$\begin{aligned} Dist(U_1, U_2) =& W_1 \times |G'_{norm}(U_1) - G'_{norm}(U_2)| \\ &+ W_2 \times L_{2,norm}(K(U_1), K(U_2)) \\ &+ W_3 \times L_{2,norm}(A(U_1), A(U_2)) \\ &+ W_4 \times L_{2,norm}(H(U_1), H(U_2)) \end{aligned} \tag{4.17}$$

where G'_{norm} is the Normalized Integral Geodesic. K, A, and H are the geometric surface vectors representing curvature, area, and average distance, respectively, and they implicitly capture different branch arrangements relative to a topological feature. Hence, G'_{norm}, K, A, and H together describe the spatial location of the topological feature. W_1, W_2, W_3, and W_4 are ratios such that $W_1+W_2+W_3+W_4 = 1$. We use these weights to adjust the relative importance of G'_{norm}, K, A, and H. We can now avoid slow graph matching algorithms by converting the matching problem to a flow and transportation problem.

We compute W_1, W_2, W_3, and W_4 on a small dataset and exhaustively try every possible combination that sum to 1 by incrementing each of these weights by 0.01 step. We use $W_1 = 0.1$, $W_2 = 0.18$, $W_3 = 0.36$, and $W_4 = 0.36$ for all our experiments throughout this thesis.

4.6.3 Indexing Scheme

A search engine should return results accurately and within an acceptable period of time. As most users are only interested in the first few tens of returned results, most search engines would employ an indexing structure so that relevant information can be retrieved without the need to traverse the whole database. For content-based retrieval systems, this is particularly important as the database is generally very large. One of the general approaches is to define features as k-dimensional (k-d) points and apply existing spatial indexing methods, like R-tree and Kd-tree for fast

retrieval. However, as explained earlier, our features are complex and it is difficult to transform them into k-d points while preserving their distances. To take advantage of the metric similarity measure, we apply the vantage point (VP) tree [64] to construct an indexing structure here.

The VP-tree is similar to the Kd-tree in that both partition the metric space into separate spaces and build the search tree hierarchically on these spaces. While the Kd-tree chooses the median as the separating point by projecting data to a dimension axis with maximum spread, the VP-tree partitions the space based on relative distances between data points and a particular vantage point. As shown in Figure 4.17, the VP-tree algorithm chooses a vantage point vp and partitions the feature space by a radius u. The space inside the circle represents features that are at most u distance away from vp, whereas the space outside represents features that are at least u distance away from vp. A VP-tree can then be constructed with the left branch storing features inside the circle ($Space_1$) and the right branch storing features outside the circle ($Space_2$). The partition process progresses recursively on the space containing all pairwise distances between all models in the database.

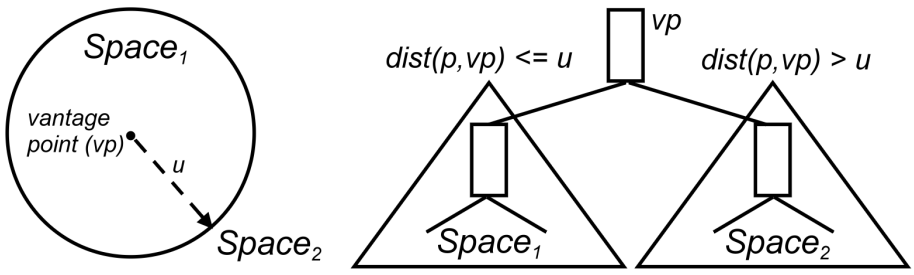


Figure 4.17: Construction of a VP-Tree

A distance-based indexing method generally requires a distance function that satisfies metric properties. Our method is based on the EMD framework, which can be proven a true metric if it satisfies the following properties under EMD formulation [86]:

1. The sum of all feature weightings for each model should be the same.

2. The ground distance function used by EMD must be a metric.

Our algorithm satisfies the first property because we use the normalized *Effective*

Area as the weights, as shown in Eqn 4.14. The ground distance $Dist()$ (Eqn 4.17) is a metric because it is a combination of metrics with positive weights that sums to 1. Such metric property has been discussed in [91] (P.187) as *Convex Combination of Metrics*.

To search for the most relevant models with respect to an input query, it is equivalent to performing a k-Nearest Neighbour (kNN) search on the VP-tree. According to [64], kNN search on the VP-tree is similar to tree traversal. It avoids unnecessary walks in the tree and so speed up search. Given a query q, as shown in Figure 4.18, a kNN search is to find all neighbors within distance l, where l is dynamically adjusted to the distance of the kth nearest neighbor. Considering query q_1, since the query space does not overlap with $Space_1$ of vantage point vp, the traversal of left side of the tree can be avoided. Similarly for q_2, since there is no overlap between the query space and $Space_2$, the traversal of right side of the tree can be avoided. This kind of pruning can significantly reduce the computational and disk-IO costs. For query q_3, where the query space overlaps with both $Space_1$ and $Space_2$, the search traverses both branches of the tree.

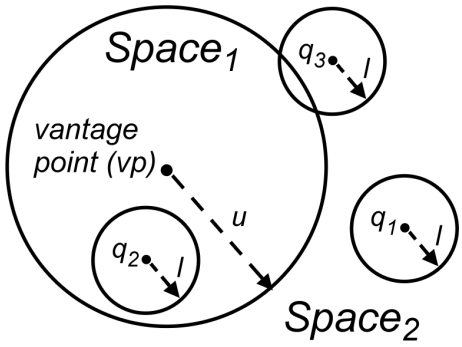


Figure 4.18: k-NN Search on a VP-Tree

4.7 Experimental Results

To evaluate the performance of the proposed retrieval method for articulated models, we discuss a number of experiments here. We have constructed a database from 150 models for these experiments. We create our own database because by our time of testing, most available dataset are small [4], or are not meshes [27]. Our database

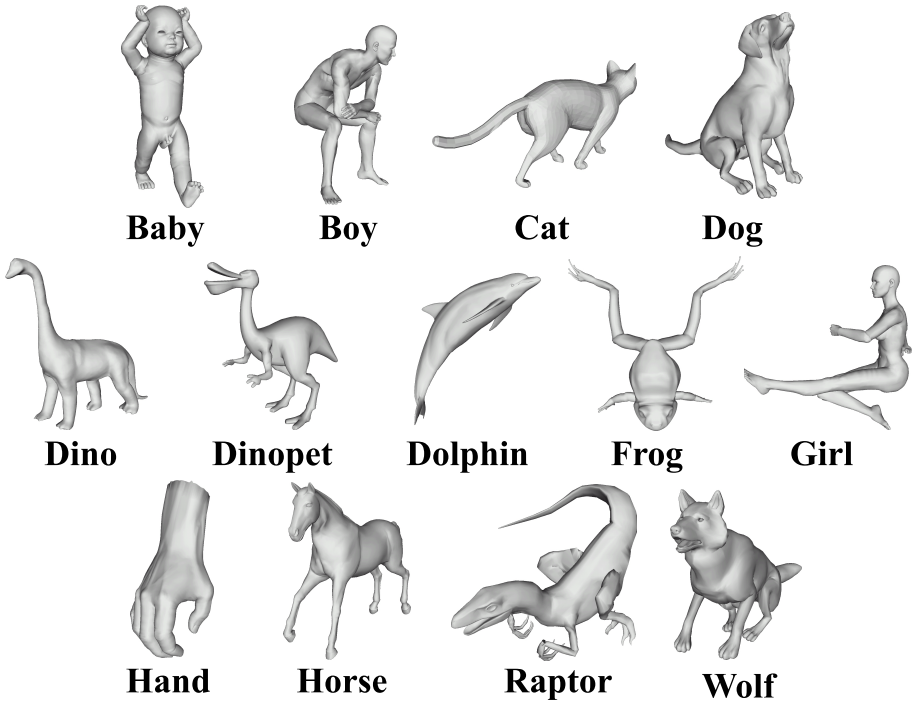


Figure 4.19: 13 groups of models in the database.

also contains lots of similar skeleton models as well. To test and compare the invariant properties of our method in rotation and scaling with other methods, we have created three additional sets by rotating the 150 models against the xy-axis, random scaling between (1.0, 2.0], and rotating by the yz-axis plus random scaling to produce a total of 600 models. We then manually categorize these models into 13 groups as shown in Figure 4.19. Each group consists of similar models but at different postures. All the experiments presented in this section are performed on a PC with a Pentium 4 2.4-GHz CPU and 1-Gbyte RAM. We use C++(Cygwin) for all our coding works. It should be noted that, for the 3 rotated and scaled copies of the same model set, our similarity measure provides nearly zero (< 0.005) values among them as our features are rotation and scaling invariant.

4.7.1 Performance Comparison

4.7.1.1 Performance on Model Discrimination

Tables 4.1 and 4.2 show some matching results, using all models and normalized by maximum and minimum work done. We can see that our method can distinguish

	boy	frog	dolphin
boy	1	0.414	0.002
frog	0.414	1	0.008
dolphin	0.002	0.008	1

Table 4.1: Mean Similarity of Dissimilar-Skeleton Models

	boy	girl	baby
boy	1	0.733	0.553
girl	0.733	1	0.458
baby	0.553	0.458	1

Table 4.2: Mean Similarity of Similar-Skeleton Models

models based on their skeletons and shapes. In Table 4.1, boy, frog, and dolphin have dissimilar skeletons. Our method can discriminate them as the similarity values among different model groups are relatively small. In Table 4.2, boy, girl, and baby are model groups with similar skeletons. Our method again can discriminate them as the similarity values among different model groups are still comparatively small, although they are slightly larger than those in Table 4.1. These two sets of results match human perception well.

4.7.1.2 Performance Comparison with Non-Articulated Methods

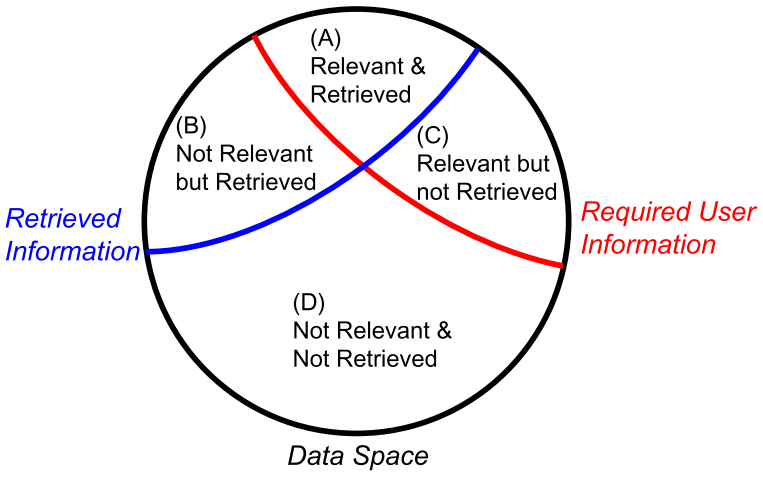


Figure 4.20: Precision and Recall

To evaluate the performance of our method, we use Precision and Recall. It measures the reliability of a system. Precision is defined as $\frac{\text{relevant retrieved data}}{\text{retrieved data}}$. Recall is defined as $\frac{\text{relevant retrieved data}}{\text{relevant data}}$. Increasing the pool of returned data may increase the chance of finding user's need, but at the same time, it may also return lots of irrelevant data. If Precision and Recall are both high, it means the system can fit the user's need. To define relevancy, a set of predefined categories have to be given. In Figure 4.20, we have illustrated how Precision $\frac{A}{A+B}$ and Recall $\frac{A}{A+C}$ are defined. In summary, the more the curve approaches the top-right hand corner, the more accurate is the method.

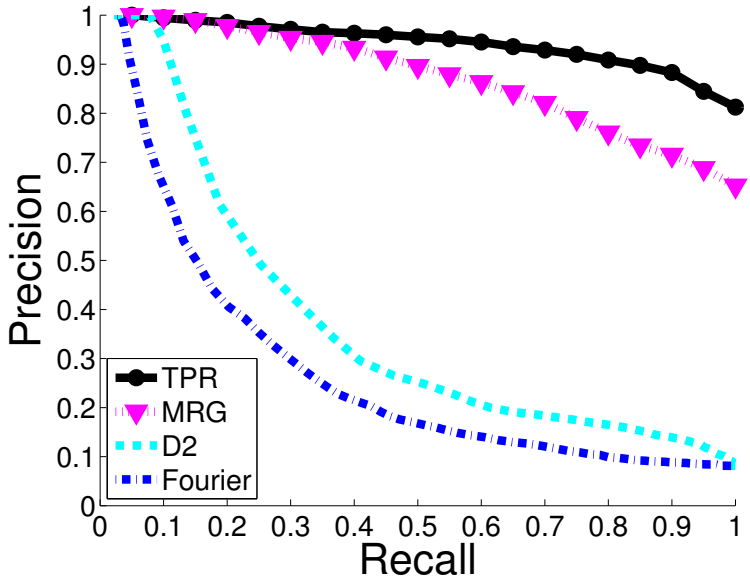


Figure 4.21: Performance Comparison of D2, Fourier, MRG and TPR

Figure 4.21 shows the precision-recall graph of our method compared with some non-articulated methods. It shows that our method outperforms the geometry-based D2 method [27] (feature size: 72) and the transform-based Fourier method [32] (feature size: 21). From the plot, we may conclude that our method is capable of handling articulated models, whereas the D2 and Fourier methods are not as their precisions drop dramatically when the recalls rise over 0.1.

4.7.1.3 Performance Comparison with MRG (Articulated Method)

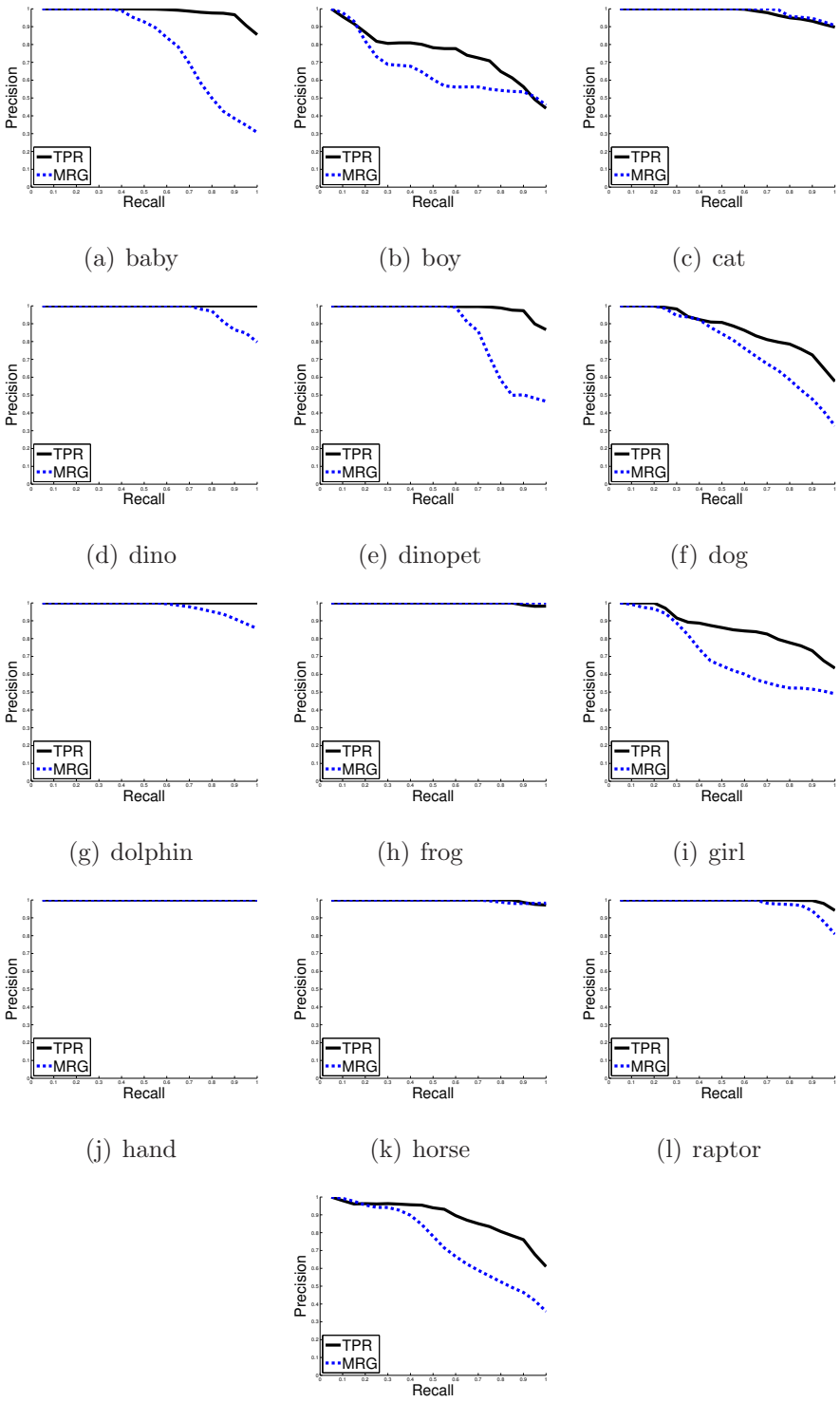
Methods for articulated geometry methods are generally difficult to implement as they require specific skeletal tree construction and graph matching techniques. To study the performance of the new method as compared to existing ones, we have implemented the MRG method [17] for comparison. We have chosen to implement MRG because both MRG and TPR make use of geodesic distance. Also, both of these methods have the largest number of features. We compare the performance of TPR with MRG here.

Accuracy Comparison We have two observations in Figure 4.21. First, both TPR and MRG can handle articulated geometry models because the precision and recall of both methods are better than non-articulated methods. Second, in the same figure, TPR outperforms MRG in the precision and recall curves when recall is above 0.3.

To explain the second observation, we may analyze the features used by TPR and MRG. For MRG, the similarity measure is based on matching multiresolution reeb graphs. With similar-skeleton models, the differences can only be captured at the lowest level of the graphs using local geometric features, area, and length. These features are local and do not represent the overall shape of the model. As a result, MRG is less effective on these models. Although TPR represents similar-skeleton models with similar number of topological features, it captures the overall shape of the models by the global geometric features and weights. For example, dog, wolf, cat, horse, and dino are four-legged animals with tails, dolphin has four side fins and a back fin, and the hand has five fingers. It is difficult to discriminate them if we consider only topological and local geometry information, as in MRG.

To further compare the performance of the two methods, we have also plotted the precision-recall graph for each model group. Figure 4.22 shows some of them. We observe that TPR performs better than MRG for model groups like dog, wolf, raptor, dinopet, baby, girl, man, dino, and dolphin or equally well as MRG for model groups like hand, frog, cat, and horse.

From these graphs, we notice that TPR outperforms MRG particularly on similar-



(a) baby (b) boy (c) cat

(d) dino (e) dinopet (f) dog

(g) dolphin (h) frog (i) girl

(j) hand (k) horse (l) raptor

(m) wolf

Figure 4.22: Precision-recall graphs for some of the model groups.

skeleton models such as canines (dog, wolf), cannibal dinosaurs (raptors, dinopet), and humans (baby, girl, man). These models have similar skeletons that cause MRG to drop in performance. By also considering the global geometric features, TPR performs significantly better. For example, in the case of baby and girl, the arms, legs, and bodies of girl are relatively longer than those of baby, and in the case of dog and wolf, the bodies and ears of wolf are fatter and sharper, respectively, than those of dog. All of these differences affect the weights and distributions of area, curvature, and average distance in TPR. We also notice that both TPR and MRG perform well on hand, frog, cat, and horse. This indicates that the range of maximum and minimum similarity values of one model group does not overlap with those of the other model groups. This is because their shapes are comparatively unique in our database. As a result, both TPR and MRG perform equally well on them.

4.7.1.4 Speed Comparison

We have also compared the time complexity of different processes between TPR and MRG as follows:

Feature Extraction Here we discuss the complexity of the feature extraction method. First, we apply LSD heuristics and hierarchical partitioning to find three source points using the Dijsktra algorithm. Second, we apply Critical Point Analysis on these three source points to identify feature points of the model. Third, we apply Topological Point Selection, which will stop as soon as the search radius is reached. Hence, only boundary vertices of each region may be visited more than once. The selection process thus visits most of the vertices, and its complexity is slightly higher than $O(n \log n)$ but bounded by $R \times O(n \log n)$, where n is the number of vertices and edges in the models. *Topological Ring Extraction* is a modified Dijsktra algorithm, and its complexity is the same as Dijsktra. For integral geodesic calculation, it is $(\mu + \gamma) \times O(n \log n)$ where μ and γ are the number of topological points and rings found. The overall complexity of the whole algorithm is $(\mu + \gamma + \psi) \times O(n \log n)$, where ψ is the number of integral geodesic calculated by hierarchical partitioning. It indicates that the algorithm depends on the number of geodesic calculations $\mu+\gamma+\psi$.

per models	TPR	MRG
Average number of vertices / triangles	9241/18478	9241/18478
Average number of integral geodesic	50	131
Average time for geodesic computation	27.51s	72.64s
Number of features	30	555
Average total time for feature extraction	45.75s	75.23s

Table 4.3: Time Analysis of TPR and MRG on Feature Extraction

Similar to TPR, MRG also requires the computation of integral geodesics for interval partitioning. It first samples a large number of seeds regularly over the model surface. It then computes the integral geodesics and interpolates the values over other vertices. In order to obtain a good approximation for interval partitioning, the number of seeds required is usually over 130 as shown in Table 4.3. TPR is comparatively much more efficient as it does not require a large number of seeds. Further, we limit the geodesic calculations to topologically important locations only as they dominate the overall feature extraction time. Hence, the number of geodesic calculations has an average value of 50. This significantly speeds up the whole process. Table 4.3 compares the performance of TPR and MRG. We can see that TPR is nearly two times faster than MRG to do the geodesic calculations. However, the overall feature extraction time of TPR is only about 40 percent faster. This is mainly due to the higher cost in computing the global geometric features.

Feature Matching We apply the EMD to compare the features of two models. A theoretical computation analysis on the complexity of EMD is difficult as it is based on the simplex algorithm. However, according to [86], if EMD is formulated as a bipartite graph problem with signatures of the same size, the time complexity is roughly $O(n^3 \log n)$ where n is the number of topological features. As a comparison, the overall complexity of MRG is $O(n_r \times (m_r + n_r))$ where m_r and n_r are the numbers of r-nodes of the two matching models. Hence, TPR has a higher complexity. However, as shown in Table 4.4, matching one model using TPR is 15 times faster than MRG because TPR has a much smaller number of topological features $n_t \approx 30$.

For MRG, the number of r-nodes, $n_r > 500$.

	TPR	MRG
Number of features	30	555
Average total time for matching two models	1ms	16ms
Average total time for one query of database	0.6s	9.6s
Total time for our retrieval application	529s	5840s

Table 4.4: Time Analysis of TPR and MRG on Feature Matching

4.7.2 Performance of the Indexing Scheme

We have created a VP-tree with two fanouts. It stores at most two data points in each node. To build the indexing tree for our existing database, it takes 778.7 s. To carry out the retrieval test, we use all models in the database as input queries and vary k for nearest neighbor search. Table 4.5 shows the total time of the k-Nearest Neighbor Search (using all 600 models as input queries) against k (the number of returned models) in the indexing scheme. We can see that if we perform a similar experiment as in Table 4.4 (that is, $k = 600$), the total time required (531.2 s) is roughly the same as that in Table 4.4 (529.1 s). However, as most users typically want to retrieve only a few models that are most relevant to a given query, the indexing scheme would certainly speed up the retrieval process. In the extreme case, if a user just wants to find the best match, the retrieval system can handle 1-nearest neighbor search in 0.39 s on average, which is 44 percent of the original full matching time of TPR without indexing or 4 percent of the matching time of MRG. Table 4.5 also shows that the total computation time is proportional to the total number of EMD operations. It should be noted that the query results are directly returned to the user in one step, without extra geometric pruning step.

4.7.3 Discussion and the Curse of Dimensionality

From the above experiments, we may conclude that TPR outperforms MRG in both accuracy and speed. Our feature representation (Topological Points and Rings)

Indexing			Average Query Time		
k	♯ EMD	Total Time	TPR(k-NNS)	TPR	MRG
1	261	233.56s	0.39s	0.88s	9.73s
2	282	250.75s	0.42s		
3	315	284.06s	0.47s		
4	357	322.95s	0.54s		
5	479	439.50s	0.73s		
6	483	444.00s	0.74s		
7	489	448.64s	0.75s		
8	492	450.13s	0.75s		
9	504	458.48s	0.76s		
10	505	459.16s	0.77s		
25	536	480.86s	0.80s		
600	600	531.22s	0.89s		

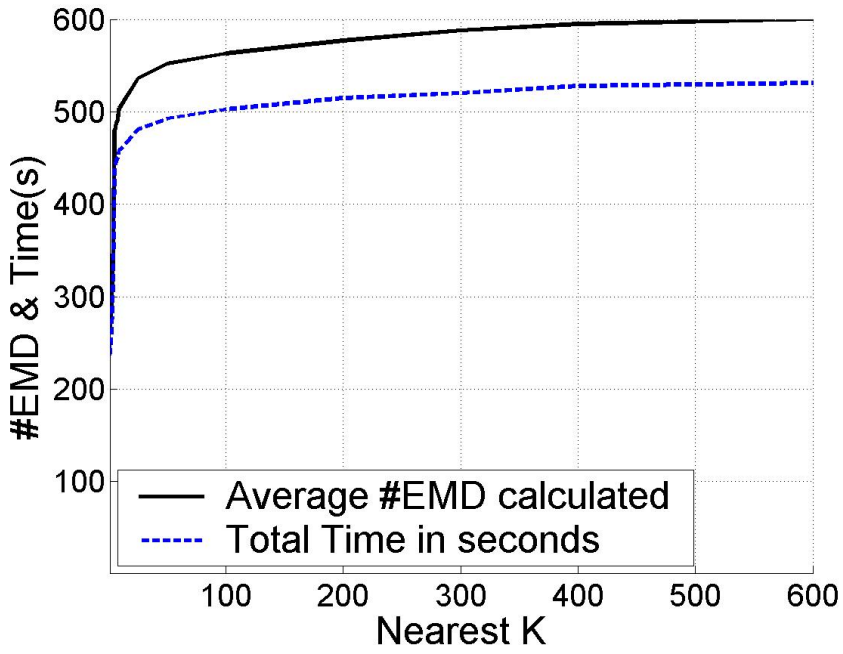


Table 4.5: Summary of k-Nearest Neighbor Search

is descriptive to distinguish both similar skeleton and dissimilar skeleton models. However, we also notice several problems. First the testing database of 600 models is small. These models are all generated by rotation and scaling from 150 models. Second, though our method supports indexing, the algorithm soon approaches brute-force when $k > 10$ (Table 4.5). This means that if a user wants to search for more data, it would be very slow. We suspect that it is due to the problem of high dimensionality of our features (≈ 900). The Curse of dimensionality is a notorious problem in multimedia retrieval systems. Third, though we have used a lot of geometric features, most of the groups still suffer from precision drops at high recall. In Chapter 5, we examine these problems in detail and propose an embedding retrieval framework to handle these data.

4.8 Conclusion

In conclusion, we have introduced a novel and efficient method for retrieving 3D articulated geometry models in this chapter. Unlike existing methods, we propose to use topological points and rings to describe each 3D model. By using additional global geometric features and weights to describe the importance of features, the matching can be modeled as a flow and transportation (EMD) problem. This opposes traditional methods that require skeletal or graph matching algorithms for matching topological entities. Our experimental results show that the new matching method outperforms MRG [17]. In addition, since our similarity measure is a metric function, our method allows indexing techniques to be applied. This not only speeds up the search process, but is also the first method that indexes both topological and geometric information in a single search.

To complete the task, we have also implemented a prototype web interface (Figure 4.23) for the retrieval system. The interface allows users to upload a model, extract, match features, search and return a list of relevant models. The returned models are sorted in descending order of similarity values.

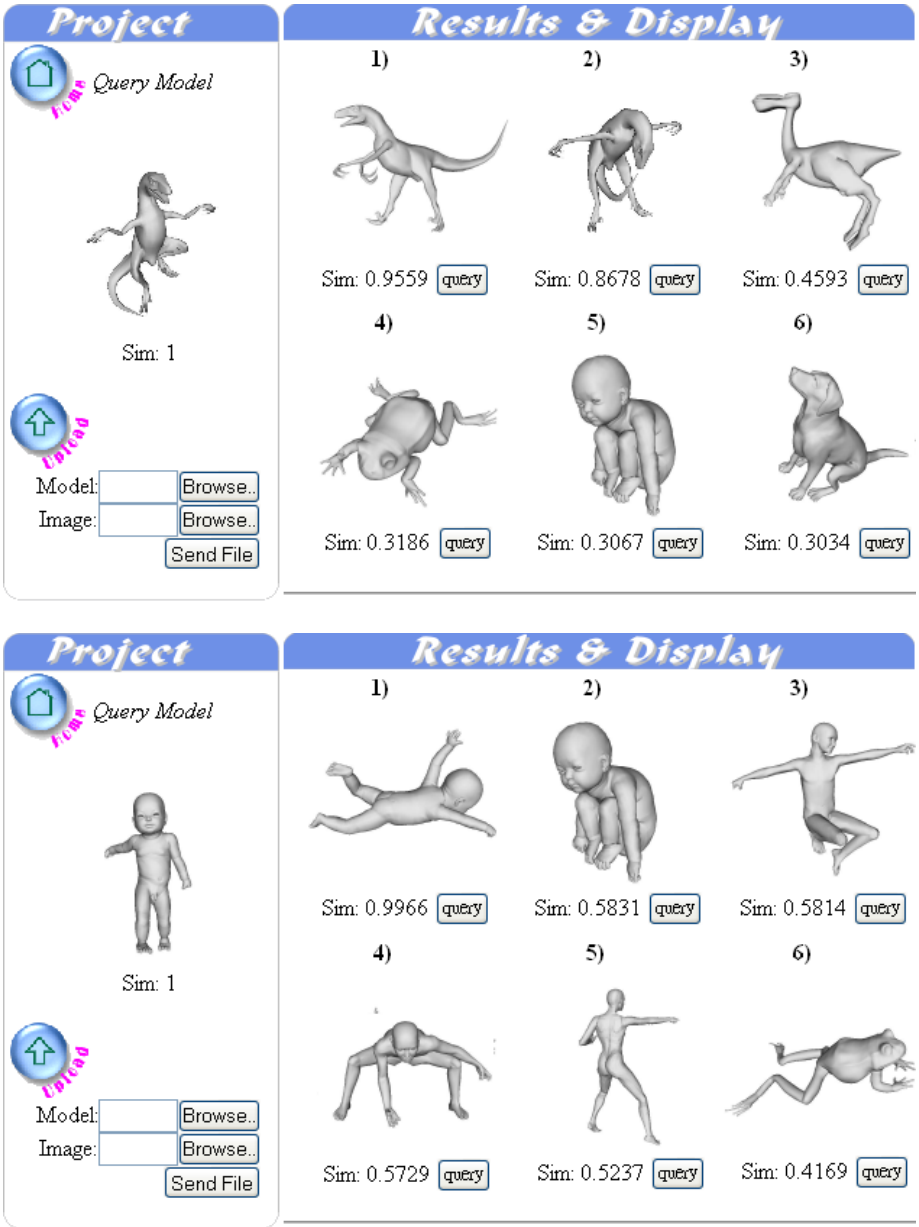


Figure 4.23: Interfaces for the Retrieval System

Chapter 5

Embedding Retrieval of 3D Articulated Geometry Models

5.1 Introduction

In the last chapter, we have developed a feature extraction and feature matching method. Though the method is a metric similarity measure and supports indexing, there are some questions unanswered.

First of all, recent works of graph-based or bag-based matching methods [10, 13, 17] propose to incorporate more descriptive geometric features into the matching process. From our experiments and observation, however, this may not improve the retrieval accuracy significantly. On the contrary, it may lead to sudden drop in precision at high recall, which is one of the observations that we have had from the results of the previous chapter.

As an illustrative example, the top diagram of Figure 5.1 shows the pairwise distances of TPR, which uses a large number of geometric features (≈ 900). These distances are measured between two sets of similar skeleton models: dog and wolf. When projected using Multi-Dimensional Scaling, these models all lie on different but nearby manifolds in the embedding space. These manifolds are so close to each other that the intra-class variance is usually greater than the inter-class distance. Retrieval using these methods will fail especially when a large number of similar skeleton models, e.g., cat, lion and horse, are in the database. This contradicts

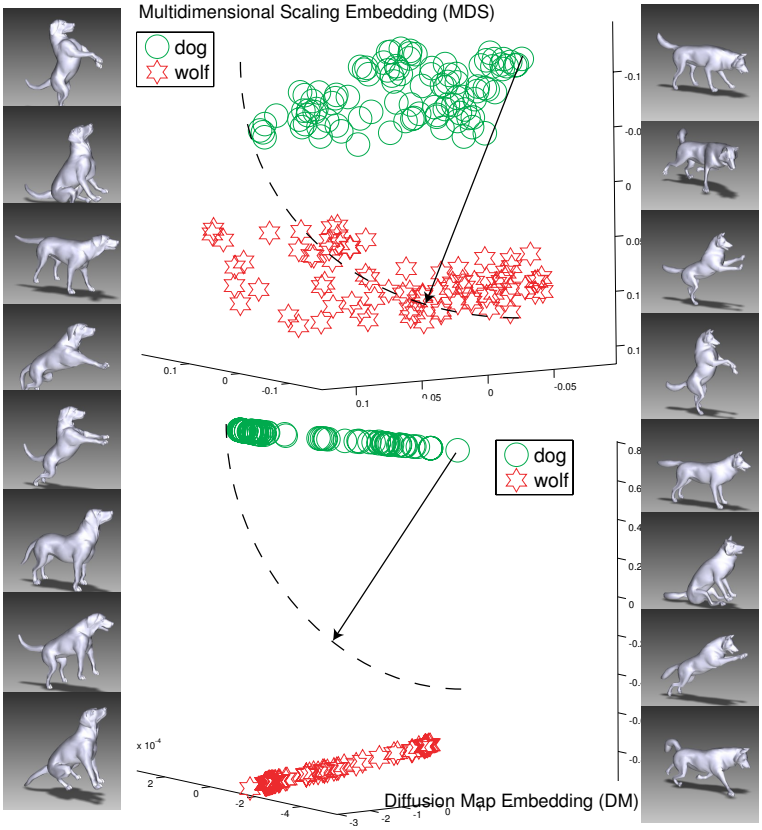


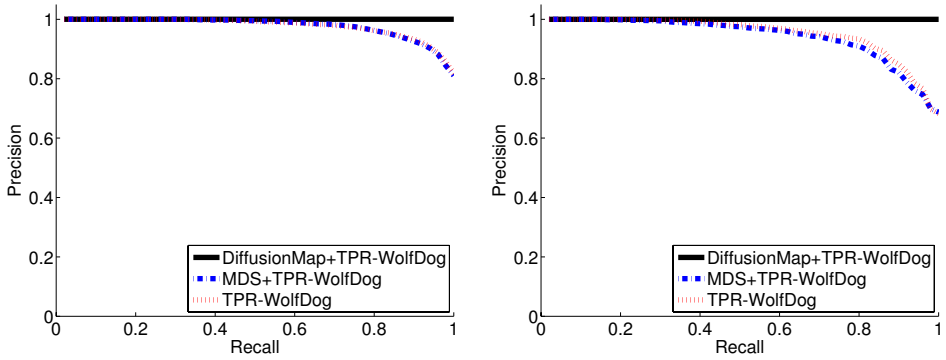
Figure 5.1: Method overview: Incorporating more geometric features may not necessarily improve the retrieval accuracy. Top: visualizing pairwise distances using MultiDimensional Scaling. Intra-class variance is larger than inter-class distance. Bottom: visualizing pairwise distances using our method. Inter-class distance is maximized.

---

with the general assumption that increasing the number of geometric features can improve retrieval accuracy. This gives rise to two questions. Why do they lie on manifolds with large variance and how can we improve retrieval accuracy in such situation?

The second problem is that these methods do not scale well to large databases. On the one hand, these methods define a similarity measure using graph-based or bag-based matching, which is non-metric and cannot be used with traditional indexing techniques. On the other hand, all these methods have high dimensionality, which greatly degrades retrieval efficiency. As an example, the retrieval speed of

TPR approaches to brute-force when the number of returned models is higher than 10, even though the similarity measure is a metric and the method uses a distance-based indexing technique, VP-Tree, for k-nearest neighbor search.



(a) Dog (b) Wolf

Figure 5.2: Precision and recall comparison among methods without embedding, and embedding retrieval by MDS and by our method. Our method can achieve very high precision (full recall) in this example.

To proceed, we first answer the question why our method projects data on manifolds. To improve the reliability and efficiency, we propose an unsupervised embedding retrieval framework for articulated geometry models. The method is based on a manifold learning technique, Diffusion Map [92], which carries out dimension reduction and maximizes inter-class distances in the induced embedding space as shown in the lower diagram of Figure 5.1. The black lines in Figure 5.2 show that the retrieval results are significantly improved. Since the space is of low dimension, spatial indexing techniques, such as kd-tree, can be applied here for fast retrieval.

However, the manifold learning approach may fail if the same group of data lie on disconnected manifolds due to insufficient objects in the database, instability of the features, or instability of the similarity measure. Such problems are usually corrected by a supervised or semi-supervised solution. As our objective is to have a fully automatic retrieval scheme, we propose to augment the kernel matrix using another similarity measure. Our argument is that two similar models may not be considered similar in one similarity measure, but may do so in another. By combining the two similarity measures, shortcut edges connecting disjoint manifolds can be established automatically to improve retrieval results.

Our retrieval framework has adapted weighted-Nyström extension to extend embedding to large databases. Although the quality of the extended embedding improves as the number of landmarks increases, increasing this number affects retrieval speed. To overcome this problem, we need a way to preserve retrieval accuracy. We have observed that the true embedding ψ obtained from eigensolver of all objects O in the database and the extended embedding $\hat{\psi}$ obtained from a few landmarks objects $\hat{O} \subset O$ are two *similar* but *distinct* embeddings. Our idea to preserve retrieval accuracy is to use ψ on database objects. To enable fast online query search, we compute the embedding coordinate $\hat{\psi}(q)$ for each new query q using Nyström extension. To relate $\hat{\psi}$ and ψ, we align them through correspondence analysis. Such novel scheme is robust and can effectively reduce both projection and retrieval errors.

We have tested the framework on both TPR (metric) and MRG (non-metric). Our experimental results show a 20-30% improvement in precision at high recall and 3-5 time improvement in speed. It also avoids the curse of dimensionality.

To the best of our knowledge this is the first comprehensive empirical study on the use of manifold learning methods in the context of retrieval and indexing of 3D articulated geometry models. We summarize our contributions as follows:

1. We reason that existing matching methods project data on nearby manifolds, and explain that when intra-class variance is greater than inter-class distance, retrieval accuracy is degraded. To address this problem, we propose an embedding retrieval framework based on the Diffusion Map.

2. To handle the disjoint manifold problem, we augment the kernel matrix with shortcut edges using another similarity measure. This is novel compared to existing works that involve supervised or semi-supervised learning.

3. To adapt weighted Nyström extension for the computation of diffusion embedding, we propose an efficient step to separate distribution from geometry. As shown in our experiment, it gives a better diffusion embedding than simple Nyström extension.

4. We propose a correspondence analysis to align query coordinate from approximated embedding into true embedding to reduce the retrieval error due to

approximation. Such an alignment step is applicable to all other retrieval schemes using Nyström extension for retrieval (e.g., landmark MDS [93]).

The rest of the chapter is organized as follows. Section 5.2 explains why TPR and MRG project 3D models on manifolds. Section 5.3 discusses the background and the use of Diffusion Map, and Section 5.4 discusses our proposed framework and Nyström extension for retrieval. Section 5.6 evaluates the performance of our retrieval framework through a number of experiments. Finally, Section 5.7 briefly concludes this work.

5.2 Manifolds in Embedding Space

All of the methods that we have discussed work well on dissimilar skeletons models. When two models of similar skeletons are matched, the best way to tell them apart is by using geometric features because most of the skeletal / topological features are likely the same. As such, the general idea is to use more geometric features for comparison. To test this hypothesis, we have created a database of 1,020 articulated geometry models featuring many similar and dissimilar skeleton models. To simulate the effect of large databases, we generate all these models by exporting each frame of some animation sets. The reasons for using animation sets are that 3D articulated geometry models are frequently used in animation sequences and they typically have a limited number of poses. This also ensures that all models are different from each other and gives a more fair evaluation than simply rotating and scaling models as in the last chapter. After obtaining this database, we compute and embed all distances using Multi-Dimensional Scaling (MDS). MDS is a popular visualization tool for preserving all pairwise distances. We have tried our database on two methods, TPR and MRG, which has the highest number of geometric features (dimensions are 800-900 approximately), as shown in Figure 5.3.

We have two observations. First, these models form many nearby clusters. These clusters have some non-linear structures (e.g., manifolds) as embedded in the Euclidean space. Many of these models have neighborly relationship with one another. We use the term “manifold-like” in our context from here. Second, the intra-class

variances (the spread of the structure) are larger than inter-class distances (the gap between two clusters) among similar skeleton models. The second observation directly accounts for the fact that the retrieval accuracy drops at high recall. This contradicts with the assumption that using more geometric features can improve retrieval accuracy for similar skeleton models.

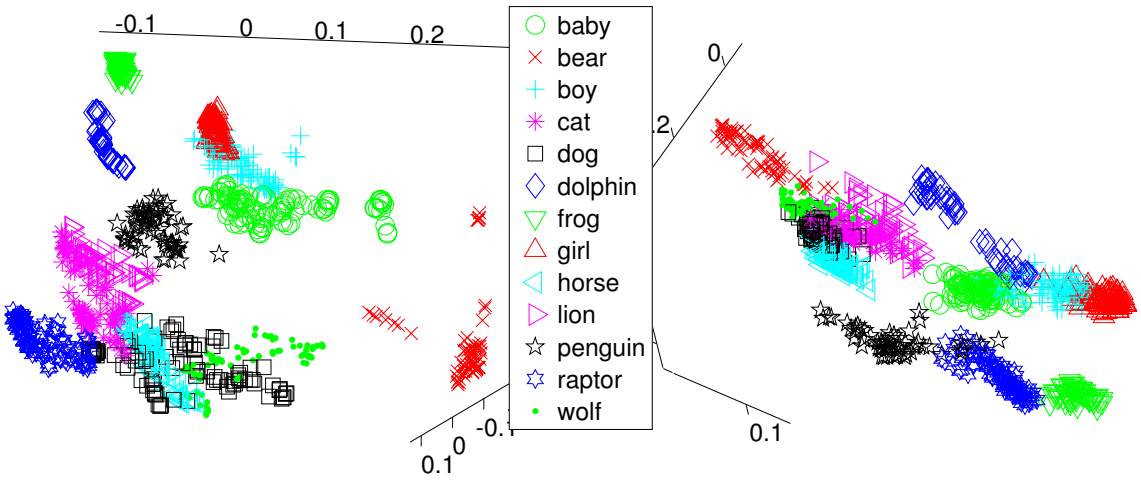


Figure 5.3: MDS visualization of (Left) a graph-based method and (Right) a bag-based method – similar skeleton models may not form separable clusters.

To explain these two observations, we first analyze the features used in these methods. MRG [17] is a graph-based method. It uses integral geodesic (centricity) to partition a model into intervals and construct a Multiresolution Reeb Graph. TPR is a bag-based method. It uses a bag of geodesic histograms as features. Both methods design geometric features that adapt to the underlying topology and hence deformation. When two similar skeleton models are matched, the nodes or histograms of the two models should be very similar. As graph matching and bag-based matching are designed in a way to find correspondences between the two models, when two similar skeleton models are matched, such similarity measure becomes Euclidean distance of high dimension $\hat{n}\hat{m}$, where $\hat{n}$ is the number of nodes in the graph/set and $\hat{m}$ is the number of features in each node/histogram as shown in Figure 5.4. In TPR and MRG, $\hat{m} = 60$ (3 sets of surface distribution (20 features each)) and $\hat{m} = 2$ (area and length), respectively.

However, Euclidean distance is very sensitive to slight misalignment. As pointed

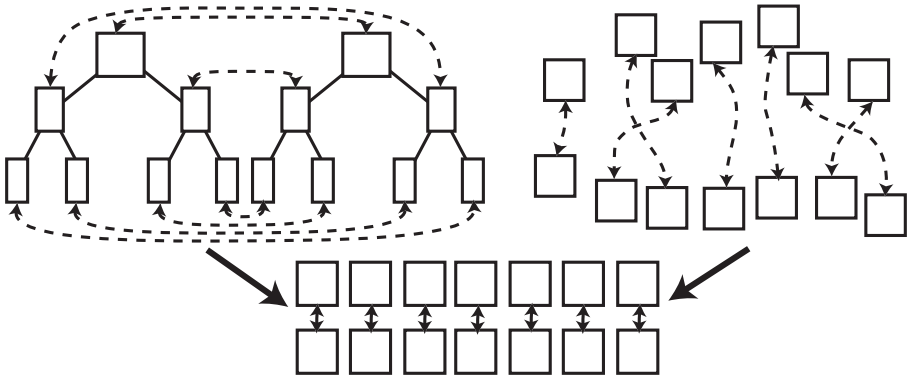


Figure 5.4: Similarity Measure and Euclidean distance. The similarity measures, which are defined by graph (upper left) and set (upper right) matching, become Euclidean distances when the 3D models for comparison have highly similar skeletons and geometric features.

out in [94] (Chapter 2), Euclidean distance is not a smooth function with respect to the natural parameters (deformation in our concern). Due to quantization, the situation may be even worse. As an illustration, consider the four model signatures shown in Figure 5.5. These histograms are obtained based on geodesic partitioning of a feature extracted from one of the legs of each animal (Section 4.5.5). We see that the histograms of all dog models have a sharp peak while that of the wolf model has a round peak. Two of the dogs are close to each other while the right dog is slightly misaligned due to articulation change. The histogram of the wolf has a peak roughly the same distance as that of the right dog. However, the computed Euclidean distances are 0.2271 (left dog & middle dog), 0.7285 (left dog & right dog), and 0.5853 (left dog & wolf). In other words, though the shape of the right dog is similar to that of the left dog, the wolf has a smaller Euclidean distance instead. This shows that misalignment may easily lead to large intra-class variance. When the variance is greater than inter-class distance, it affects retrieval accuracy. Further, since our database is generated from animation sequences, models of consecutive frames form a local neighborhood. All these explain the fact that when the database is large, they are observed as manifolds in the embedding space.

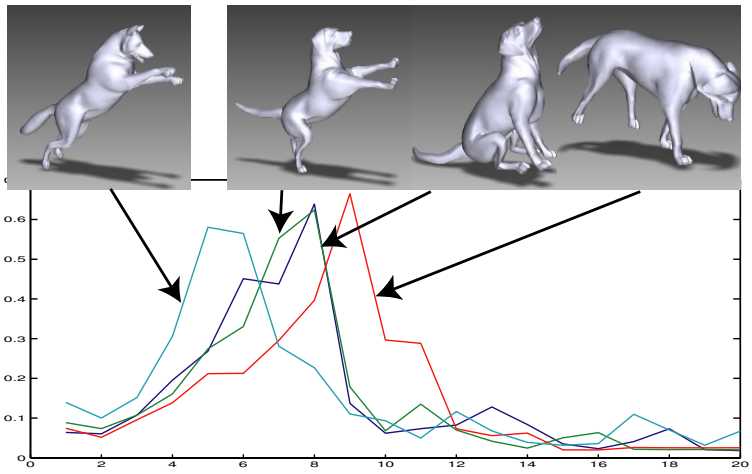


Figure 5.5: Feature histogram of 4 different models (from left: wolf, dog, dog, dog).

5.3 Embedding Retrieval and Diffusion Map

Our idea to improve retrieval accuracy is to find a new embedding such that the inter-class distances among different manifolds are maximized. This turns out to share exactly the same idea as manifold learning. In fact, applying manifold learning to better understand data is not new, e.g., [95] for image segmentation and [96] for mesh clustering. Manifold learning tools, including Multi-Dimensional Scaling [53], Local Linear Embedding [78], ISOMAP [79] and Diffusion Map [94], have also been applied to various image retrieval works, e.g., image clustering [97], relevancy feedback [98] and relevancy feedback by transduction [99].

There are two reasons that manifold learning techniques work well on all these applications. First, manifold learning techniques project pairwise distances on leading eigenvectors. According to the Polarization Theorem [100] (Theorem 5.6), the angles between eigenvectors become maximized when the projected dimensionality is reduced. In other words, the embedding distance between data is maximized in low dimensional embedding space. This provides the reasons why segmentation and clustering algorithms usually work better in the transformed domain. Second, the success of applying manifold learning on image retrieval results from the assumption that visual perception is better represented by nonlinear distance than its original distance. As we have observed in the previous section and demonstrated later in our experiments, this assumption can also be applied to 3D model features because

they also lie on manifolds.

Among various manifold learning techniques, we have chosen the Diffusion Map and the diffusion embedding space [94]. Diffusion Map has an advantage over existing manifold learning techniques that it infers far distance by dyadic power (diffusion) of local distance. It does not require explicit graph construction as in ISOMAP [79] nor explicit maximization of far distance [97], but only the values of two global parameters σ and t. This is important because finding the nearest neighbors for a new query is equivalent to sequential scanning of the whole database, which is prohibitive for fast online query search and does not scale to large databases.

Our work also differs from existing works in that we use the Diffusion Map as an unsupervised retrieval method and optimize all the parameters based on retrieval constraints. When the data lie on disconnected manifolds, we further propose to combine several similarity measures together by means of shortcut edges. Most important of all, our method does not require supervised training, although it also works nicely with the relevancy feedback approach [99] as the kernel is fixed once optimization is done. It should also be noted that manifold learning techniques are usually applied on image retrieval with single feature vector representation. In this work, we have showed evidence that they are also applicable to graph and bag-based matching methods, in particular, TPR and MRG, where the features are extracted according to some surface metrics.

To allow a more complete view of our framework, we first briefly summarize the Diffusion Map. Let $W(x, y)$ be the pairwise distance matrix obtained by graph-based / bag-based methods. Then we compute a kernel matrix $K_w(x, y)$:

$$K_w(x, y) = \exp\left(-\frac{W^2(x, y)}{\sigma}\right) \tag{5.1}$$

where σ is a parameter that defines the local scale of the neighborhood and $\exp()$, exponential function, is applied entry-wise to the distance matrix W. The use of an exponential function suggests that small distances are important while large distances are ignored. This is essential to learning the manifolds because they are defined by local neighborhoods. Since these data may have different distributions in various points, it is best to separate distribution from the geometry so that the

embedding is not affected by local factors. Such distribution can be estimated by letting $p_w(x) = \sum_{y \in O} K_w(x, y)$, where O represents all the objects in the database. Then, the kernel matrix $K(x, y)$, which has the distribution separated from geometry, can be defined as:

$$K(x, y) = \frac{K_w(x, y)}{p_w(x) p_w(y)} \tag{5.2}$$

However, this kernel $K(x, y)$ is not symmetric, and a graph normalization technique is usually adopted to symmetrize it as follows. Let $q(x) = \sum_{y \in O} K(x, y)$. The symmetric anisotropic transition kernel $P(x, y)$ of Markov chain on O is defined as:

$$P(x, y) = \frac{K(x, y)}{\sqrt{q(x)}\sqrt{q(y)}} \tag{5.3}$$

The diffusion distance $D_t(x, y)$ of the embedding space is defined as:

$$D_t^2(x, y) = \sum_{u \in O} \frac{P(x, u) - P(y, u)}{\pi(u)} \tag{5.4}$$

where $\pi(u) = q(u)/\sum_{z \in O} q(z)$ is the stationary distribution of the Markov chain. Let d be the dimension of the embedding space, the diffusion distance $D_t(x, y)$ can be approximated using right eigenvectors ψ and eigenvalues λ of $P(x, y)$:

$$D_t(x, y) = \left(\sum^{d} \lambda^{2t} \left(\psi(x) - \psi(y) \right)^2 \right)^{\frac{1}{2}} \tag{5.5}$$

The Diffusion Map $\Psi_t(x) : O \rightarrow \mathbb{R}^d$ thus embeds all 3D models into an Euclidean space.

$$\Psi_t(x) = \left(\lambda_1^t \psi_1(x), \lambda_2^t \psi_2(x), ..., \lambda_d^t \psi_d(x) \right)^T \tag{5.6}$$

In this space, intra-class distances among different manifolds are maximized. Since it is in Euclidean space, we can apply a spatial indexing method (e.g., the kd-tree) for fast retrieval.

As a note, Diffusion Map also has a close relationship with Kernel Principle Component Analysis (Kernel PCA). PCA is a useful too for analyzing data and dimension reduction. However, PCA cannot handle non-linear data sets. To handle

such non-linear structure, many methods emerged from the field of manifold learning to address the issue. Important works include Local Linear Embedding (LLE) [78], Laplacian eigenmaps [101] and Hessian eigenmaps [102]. In [103], it was noted that all these methods, including Diffusion Map [94] are subcases of kernel PCA, that minimize local distortion of neighboring elements around every point.

5.4 Proposed Retrieval Framework

In retrieval literature (Section 2.4.2.2, Approximate Method), it is popular to use Nyström extension to approximate embedding of large datasets. According to [75], FastMap, MetricMap and Landmark MDS are all based on Nyström extension. These methods use Nyström extension to compute embedding for both the database (offline) and query (online) objects. The reason is that it is often not feasible to compute eigen-decomposition for a full (non-sparse) distance matrix. Eigen-decomposition is usually slow in large databases and requires a lot of memory. However, as shown in Section 5.4.5, using Nyström extension with a few landmarks objects for both the database and the queries will cause projection error. Such error distorts pairwise distances in the embedding space and thus affects retrieval accuracy. The problem becomes more severe when the number of landmarks is small and the number of dimensions is high.

To resolve the problem, we propose to obtain a true embedding (eigensolver embedding) for database objects (offline) based on full eigen-decomposition as shown in the right diagram of Figure 5.6. This is possible in our proposed framework because our kernel matrix is sparse. To compute query coordinate (online), we use the Nyström extension as well. However, we project the approximated query coordinate back to the true embedding using correspondence analysis during online query search. This gives us very accurate query coordinate for fast and reliable retrieval. In Figure 5.6, there are several blocks highlighted as dotted boxes. The diagram shows the differences between the proposed framework and existing works.

In the following subsections, we present the framework in detail. Section 5.4.1 discusses an automated algorithm to compute parameters for the Diffusion Map.

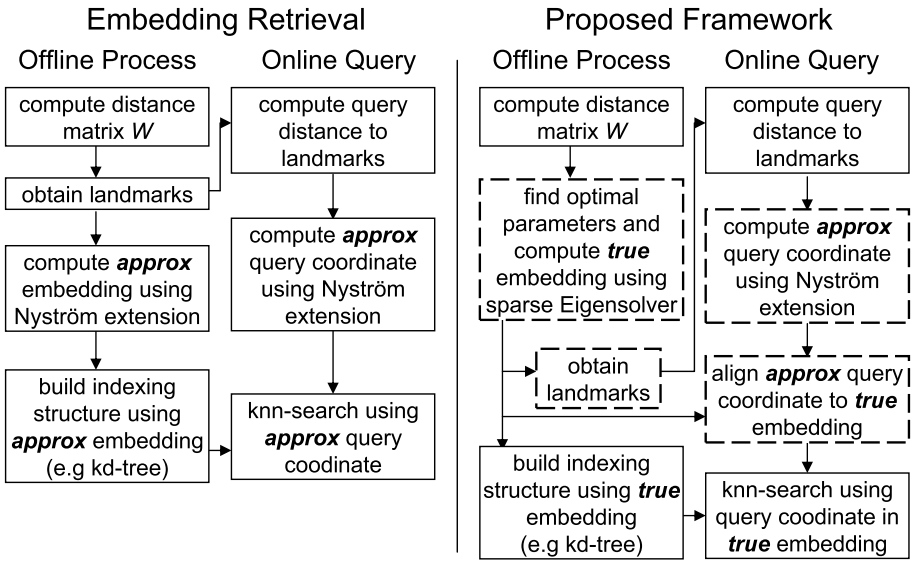


Figure 5.6: Retrieval Framework. Left: Nyström extension used in existing work. Right: Our proposed framework using both embedding from Eigensolver and Nyström extension. The dotted boxes indicate the differences.

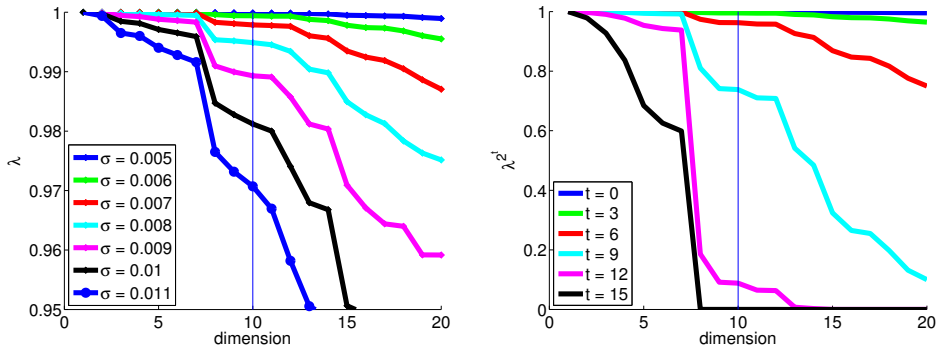
Section 5.4.2 discusses how to augment the kernel by one or more similarity measures in the form of shortcut edges. Section 5.4.3 defines a reliable weighted Nyström Extension particularly for the Diffusion Map by separating distribution from geometry. Section 5.4.4 discusses how to speed up Nyström for online query search. Section 5.4.5 discusses how to obtain a true query coordinate by correspondence analysis. Finally, Section 5.4.6 presents the algorithm to select landmarks automatically.

5.4.1 Optimizing Parameters by Retrieval Criteria

In our retrieval framework, there are three parameters to optimize: σ, t and d. Optimizing parameters for spectral algorithms is a difficult task. More often, the parameters are data-dependent and vary across different types of data. While this task is important for clustering and segmentation algorithms, no work has discussed how to optimize them for retrieval. In this subsection, we discuss the retrieval criteria and the optimization algorithms. We expect these parameters to be applied globally to the whole database.

In Diffusion Map, σ is a global parameter used to define the kernel matrix, K_w. It represents the local scale of the neighborhood. It is proven that when

$\sigma \rightarrow 0$, the kernel approaches the Laplace-Beltrami Operator. This produces smooth eigenvectors that are good for harmonic extension. A very small σ, however, results in high multiplicity of 1 in the spectrum (list of eigenvalues). From spectral graph theory, multiplicity of 1 counts the number of disconnected components (manifolds in our case). The group information is stored in the associated eigenvectors. On the one hand, this is good because it maximizes the distance between different manifolds. On the other hand, if the multiplicity of 1 is greater than d, we lose important eigenvectors and so grouping information in the embedding space. Therefore, it imposes a constraint that the optimized σ should give the largest multiplicity of 1 that is less than dimension d. For example, in Figure 5.7(a), σ is set to 0.006.



(a) σ, λ vs d (b) t, λ vs d, $\sigma = 0.006$

Figure 5.7: Relationship between d (dimension) and (a) σ (local scale), (b) t (dyadic power for diffusion).

Parameter t is the dyadic power to diffuse local distance to infer far distance. It holds the key to dimension reduction as shown in Figure 5.7(b). From the signal processing point of view, summation of spectrum represents the total energy. It would be desirable to have all the energy concentrated at the first few d dimensions so that all the important information is well-represented. Since noise is usually located at high frequencies, it may be desirable to truncate high frequency components. Due to these reasons, we compute t by defining $\xi = {\lambda_d}^{2^t} / {\lambda_1}^{2^t} = 0.1$ to truncate noise, where λ_d is the d-th eigenvalue of the spectrum. Figure 5.7(b) shows the effect of t on the spectrum with $\sigma = 0.006$. $t \approx 12$ achieves $\xi = 0.1$.

So far, σ and t are both dependent on d. In general, the higher the dimension is, the better it is to preserve pairwise distances. However, spatial indexing struc-

tures usually suffer from the curse of dimensionality. The curse degrades retrieval efficiency if the dimension is too high (> 10). Since we are using kd-tree as the indexing structure, we follow the general approach to set $d = 10$.

Algorithm 1: Optimizing parameters σ and t .

Input: d

Output: σ and t

1. compute ε_{avg}, ε_{max}, ε_{min}, which are the average, maximum and minimum, respectively, of all 1st nearest neighbor distances.
2. compute $\sigma_\varepsilon \leftarrow 0.01 \times (\varepsilon_{max}^2 - \varepsilon_{min}^2)$ as the search step size
3. initialize $\sigma \leftarrow \varepsilon_{avg}$
4. find the smallest σ by iteratively updating $\sigma \leftarrow \sigma \pm \sigma_\varepsilon$, such that $\lambda_d < 1$, by sparse eigensolver.
5. refine σ further by iteratively updating $\sigma \leftarrow \sigma \pm \sigma_\varepsilon/10$
6. compute $t \leftarrow log\left(\frac{log\xi}{log\lambda_d}\right)/log2$

Combining all these criteria, we propose Algorithm 1 to find σ and t given a predefined dimension d. The Diffusion Map embedding $\Psi_t(x)$ can then be computed according to Eq. 5.6. Since our kernel matrix is sparse, it is possible to use sparse eigensolver for direct eigen-decomposition. Though the search would still be slow for very large datasets, it is an offline process. We believe that trading off speed for accuracy here is important for reliable retrieval.

5.4.2 Augmenting Kernel Matrix

So far, we have assumed that all data lie on individual manifolds. If the features are unstable or there are insufficient samples in the database, manifolds may become disjoint. Here, we propose to directly augment the kernel matrix by means of "shortcut edges" to link these disjoint manifolds together. In general, if one method is not sufficient to discriminate two models, it is common to use another (or more) similarity measure(s) for adjustment. For example, in [4], a two-step pruning pro-

cess is used. Our idea is similar in that we use two or more similarity measures to build an automated system. However, our approach is much simpler and more efficient because it allows retrieval in one step as shown in Section 5.4.3.

We recall that kernel matrix K_w is the Markov matrix defining the probability of diffusion. With respect to spectral graph theory, a probability greater than 0 means that there is an edge connecting the two nodes. Therefore, we consider the following augmented kernel:

$$K_w = \exp\left(-\frac{W^2}{\sigma}\right) + \sum_i \alpha_i \times \exp\left(-\frac{W_i^2}{\sigma_i}\right) \tag{5.7}$$

where W and W_i are the original and the new distance matrices. σ and σ_i are the corresponding parameters obtained from Section 5.4.1. The first half of the kernel is the same as the original. The non-zero entries are the diffusion probabilities to their neighbors. By introducing the second half, we add extra probabilities, in the form of "shortcut neighbor edges", to the original kernel, where such connections may not exist. In general, such approach can be applied to more than 1 additional similarity measure. In order to reduce the negative effect of joining unrelated manifolds, α_i is introduced to limit the probability values to be added to the original kernel. Since a very small probability value is sufficient to introduce such shortcut edges, we restrict $\sum_i \alpha_i = 0.01$ in all our experiments. It should be noted that though kernel matrix K_w may now have probabilities greater than 1, it is normalized in Eqn. 5.2.

5.4.3 Nyström Extension for Diffusion Map

Multimedia retrieval usually involves large datasets containing thousands to millions records. Solving eigen-decomposition directly for online queries is infeasible due to its high computational cost. Hence approximation is usually sought. Nyström extension, as shown in Figure 5.8, is a popular technique for finding numerical approximations to eigenproblems of the form:

$$\int K(x,y)\psi(y)dy = \lambda\psi(x) \tag{5.8}$$

It approximates $\psi(x)$ by $\hat{\psi(x)}$ using quadrature rule:

$$\hat{\psi}(x) = \frac{1}{n\lambda} \sum K(x, y)\bar{\psi(y)} \tag{5.9}$$

where $x \in O$ and $y \in \hat{O}$. Eqn. 5.9 extrapolates eigenvectors (embedding) $\bar{\psi(y)}$ computed on a subset $\hat{O} \subset O$, called landmarks, to the whole database O by using distance from point x to all landmarks as weights. The theoretical support of extending the Diffusion Map using Nyström extension is discussed in [94] as geometric harmonics. However, this form of Nyström extension assumes that all landmarks have equal weights. This deviates from the integral equation that defines the kernel eigenfunctions and so requires a lot of landmarks to preserve embedding quality. In retrieval applications, we want to reduce the number of landmarks. Hence, we have adapted Density-Weighted Nyström [18] for use with the Diffusion Map.

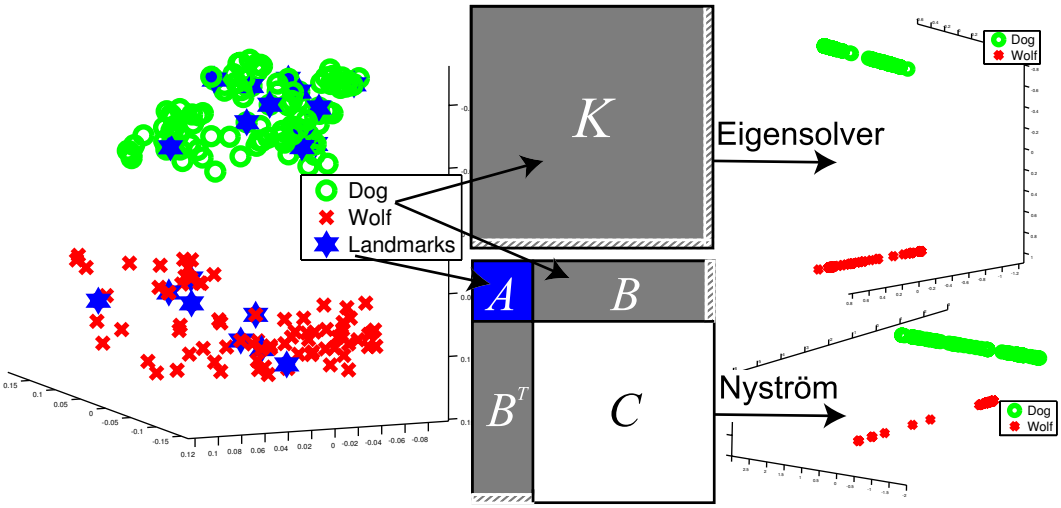


Figure 5.8: Using Eigensolver and Nyström Extension in retrieval applications. K is the kernel distance matrix. A and B are the submatrices of K. A is the distance matrix among landmarks and B is the distance matrix between landmarks and non-landmarks. The hatched regions are the distance values between a new query and the whole database (upper matrix) and between a new query and the landmarks (lower matrix). Embedding at upper right is obtained from Eigensolver (true embedding). Embedding at lower right is obtained from Nyström (approximated embedding).

Density-Weighted Nyström considered the following eigenproblem instead:

$$\int K(x,y)S(y)\psi(y)dy = n\lambda\psi(x) \tag{5.10}$$

Using $K(x,y)S(y)$ instead of $K(x,y)$ takes density $S(y)$ of landmarks into account. Since $K(x,y)S(y)$ is not symmetric, it sets $\bar{\psi}(y) = S(y)^{-\frac{1}{2}}\mu(y)$ and converts the eigenproblem into:

$$\int S(x)^{1/2}K(x,y)S(y)^{1/2}\mu(y)dy = n\lambda\mu(y) \tag{5.11}$$

where $S(x)^{1/2}K(x,y)S(y)^{1/2}$ is symmetric. This yielded a better Nyström approximation scheme:

$$\hat{\psi}(x) = \frac{1}{n\lambda}\sum K(x,y)S(y)\bar{\psi}(y) \tag{5.12}$$

Zhang et al. [18] further extended Eqn. 5.12 to solve the normalized cut problem:

$$D^{-1/2}(x)K(x,y)D^{-1/2}(y)\psi(y) = \lambda\psi(y) \tag{5.13}$$

using the Nyström approximation of:

$$\hat{\psi(x)} = \frac{1}{\lambda}\sum D^{-1/2}(x)K(x,y)S(y)D^{-1/2}(y)\bar{\psi}(y) \tag{5.14}$$

where K and D are the similarity and diagonal degree matrices, respectively.

At first glance, the Eqn. 5.14 corresponds to Eqn. 5.3 in the Diffusion Map framework (Section 5.3) and it seems that Eqn. 5.3 can be directly replaced by Eqn. 5.14 to extend the Diffusion Map. However, if we look at it carefully, Eqn. 5.3 expects a kernel matrix K that has distribution separated from geometry. Kernel K is required to compute $\bar{\psi}$, $D(x) = \sum_y K(x,y)$ and the extension itself. Separating distribution from geometry is an essential step for the Diffusion Map to obtain a true Laplace-Beltrami operator and to analyze the underlying manifold. However, in retrieval applications, we usually have weight matrix K_w (the kernel before distribution separation step), and in particular, the associated submatrices A_w and B_w only. In the following, we discuss how to obtain distribution separated K from K_w efficiently without requiring the full matrix of K_w.

First, we consider putting symmetric kernel K_w directly into the following eigen-problem.

$$\int K_w(x,y)S(y)\psi(y)dy = n\lambda\psi(x) \tag{5.15}$$

From Eqn. 5.12, the Nyström approximated eigenvectors can be rewritten in matrix form as:

$$\hat{\psi} = \begin{bmatrix} A_w \\ B_w^T \end{bmatrix} S\tilde{U}\Lambda^{-1} \tag{5.16}$$

where $\tilde{U} = \bar{\psi}$, $\Lambda = n\lambda$, $A_w = K_w(x,y) \quad x,y \in \hat{O}$ and $B = K_w(x,y) \quad x \in O - \hat{O}, y \in \hat{O}$.

Since $\hat{\psi}$ is the approximated leading eigenvectors of K_w:

$$\begin{aligned} K_w &\approx \hat{\psi}\Lambda\hat{\psi}^T \\ &= \left(\begin{bmatrix} A_w \\ B_w^T \end{bmatrix} S\tilde{U}\Lambda^{-1} \right) \Lambda \left(\Lambda^{-1}\tilde{U}^T S^T \begin{bmatrix} A_w & B_w \end{bmatrix} \right) \\ &= \begin{bmatrix} A_w \\ B_w^T \end{bmatrix} S\tilde{U}\Lambda^{-1}\tilde{U}^T S^T \begin{bmatrix} A_w & B_w \end{bmatrix} \end{aligned} \tag{5.17}$$

Let $\mu = S^{1/2}\tilde{U}$. Since $S^{1/2}AS^{1/2}$ is symmetric, we have

$$\begin{aligned} S^{1/2}AS^{1/2} &= \mu\Lambda\mu^T \\ (S^{1/2}A_wS^{1/2})^{-1} &= (\mu\Lambda\mu^T)^{-1} \\ S^{-1/2}A_w^{-1}S^{-1/2} &= (\mu^T)^{-1}\Lambda^{-1}\mu^{-1} \\ &= \mu\Lambda^{-1}\mu^T \\ &= S^{1/2}\tilde{U}\Lambda^{-1}\tilde{U}^T S^{1/2} \\ \text{and so,} \qquad A_w^{-1} &= S\tilde{U}\Lambda^{-1}\tilde{U}^T S \end{aligned}$$

By substituting it back to Eqn. 5.17, we have:

$$
\begin{aligned}
K_w \approx \bar{K_w} &= \begin{bmatrix} A_w \\ B_w^T \end{bmatrix} A_w^{-1} \begin{bmatrix} A_w & B_w \end{bmatrix} \\
&= \begin{bmatrix} A_w A_w^{-1} A_w & A_w A_w^{-1} B_w \\ B_w^T A_w^{-1} A_w & B_w^T A_w^{-1} B_w \end{bmatrix} \\
&= \begin{bmatrix} A_w & B_w \\ B_w^T & B_w^T A_w^{-1} B_w \end{bmatrix}
\end{aligned} \tag{5.18}
$$

This essentially shows that Density-Weighted Nyström has the same matrix completion view as general Nyström [104].

Suppose now we want to solve the following eigenproblem:

$$
\begin{aligned}
\int K(x,y) S(y) \psi(y) dy &= \int \frac{K_w(x,y)}{p_w(x) p_w(y)} S(y) \psi(y) dy \\
&= n\lambda \psi(x)
\end{aligned} \tag{5.19}
$$

where $p_w(x) = \sum_y K_w(x,y)$. Since we do not have the full matrix K_w, we approximate $p_w(x) = \sum_y K_w(x,y)$ by $\bar{p}_w$ as below:

$$
\begin{aligned}
\bar{p}_w(x) &= \sum_y \bar{K_w}(x,y) \\
&= \begin{bmatrix} \sum_y A_w(x,y) + \sum_y B_w(x,y) \\ \sum_x B_w(x,y) + B_w^T A_w^{-1} \sum_y B_w(x,y) \end{bmatrix}
\end{aligned} \tag{5.20}
$$

To separate distribution from geometry, that is to compute K from K_w, we compute submatrices A and B as follows:

$$
\begin{aligned}
A(x,y) &\leftarrow \frac{A_w(x,y)}{\bar{p}_w(x) \bar{p}_w(y)}, \qquad x, y \in \hat{O} \\
B(x,y) &\leftarrow \frac{B_w(x,y)}{\bar{p}_w(x) \bar{p}_w(y)}, \qquad x \in \hat{O} \backslash O, y \in \hat{O}
\end{aligned} \tag{5.21}
$$

where A and B are submatrices of $K = \begin{bmatrix} A & B \\ B^T & C \end{bmatrix}$.

After obtaining A and B, we can now use Density-Weighted Nyström extension to solve graph normalization and extrapolate eigenfunctions to the whole database.

5.4.4 Nyström Speed-up for Retrieval

Nyström extension not only avoids expensive eigensolver on large matrices, but also reduces computing the expensive similarity measures. Given an unknown query, it is sufficient to evaluate the distances between the new query and the landmarks (Figure 5.8, hatched region) to obtain the query embedding coordinate. Since the coordinate is Euclidean, a spatial indexing technique, such as the kd-tree, can be used. This is very attractive because graph / bag-based methods are usually slow and non-metric. Nyström extension thus provides a way to scale these algorithms to large databases.

To compute a query coordinate, it is sufficient to construct

$$\hat{B_w}(x,y) \leftarrow [B_w(x,y) \qquad \exp\left(-\frac{W(q,y)^2}{\sigma}\right)] \tag{5.22}$$

where $x \in O \backslash \hat{O}, y \in \hat{O}$ and $W(q,y)$ is the distance matrix (column vector) between new query q and all landmarks $\hat{O}$.

To further speed up online query searches, we can also precompute eigen-decomposition. We have observed that eigen-decomposition of submatrix A depends on our proposed distribution separation step. Given a new query, the distribution $\bar{p}_w(x)$ can be written as:

$$\begin{aligned}
\bar{p}_w(x) &= \begin{bmatrix} \sum_y A_w(x,y) + \sum_y \hat{B_w}(x,y) \\ \sum_x \hat{B_w}(x,y) + \hat{B_w}^T A_w^{-1} \sum_y \hat{B_w}(x,y) \end{bmatrix} \\
&= \begin{bmatrix} \sum_y A_w(x,y) + \sum_y B_w(x,y) + \exp\left(-\frac{W(q,y)^2}{\sigma}\right) \\ \sum_x \hat{B_w}(x,y) + \hat{B_w}^T A_w^{-1} \sum_y \hat{B_w}(x,y) \end{bmatrix} \\
&\simeq \begin{bmatrix} \sum_y A_w(x,y) + \sum_y B_w(x,y) \\ \sum_x B_w(x,y) + B_w^T A_w^{-1} \sum_y B_w(x,y) \end{bmatrix}
\end{aligned} \tag{5.23}$$

The last step is a good approximation because the addition of a column vector $\exp\left(-\frac{W(q,y)^2}{\sigma}\right)$ is negligible in large databases. The eigenproblem then solely depends on A and S only and can be precomputed offline. It is thus sufficient to extrapolate embedding for all objects (databases and queries) by simple matrix multiplication during online query search. It is a considerable speed-up especially for large databases.

Figure 5.9 shows some of the Matlab codes. We have separated the offline precomputation algorithm from the online extension algorithm. In the online section, a simple multiplication of `[A;hat_B']*S` with the precomputed eigenvector matrix `product_VE` is required to extrapolate all coordinates including the database and query objects. We have also listed the corresponding equations next to the codes for a clear understanding of our method.

5.4.5 Query Alignment

Existing works of embedding retrieval apply Nyström extension to both the database and query objects, i.e., a set of landmark objects are chosen and then the embedding coordinates of all objects in the database are computed using Nyström. To compute the embedding coordinate for a query, distances to the same set of landmarks are computed, and extended with Nyström. All these assume that the Nyström embedding gives the best approximation that does not distort retrieval accuracy. However, as seen from our experiments, this is not the case. For example, in the right diagram of Figure 5.8, the quality of Nyström embedding can be seriously affected that some coordinates of dogs are closer to those of wolves. When the nearest neighbor search is applied on this embedding, retrieval accuracy degrades as shown in Figure 5.10. This is a practical limitation of Nyström extension. Though it is well-known that increasing the number of landmarks can improve the quality of embedding, it also increases the computation of expensive similarity measures (e.g., graph / bag-based matching in our case) and significantly degrades the efficiency.

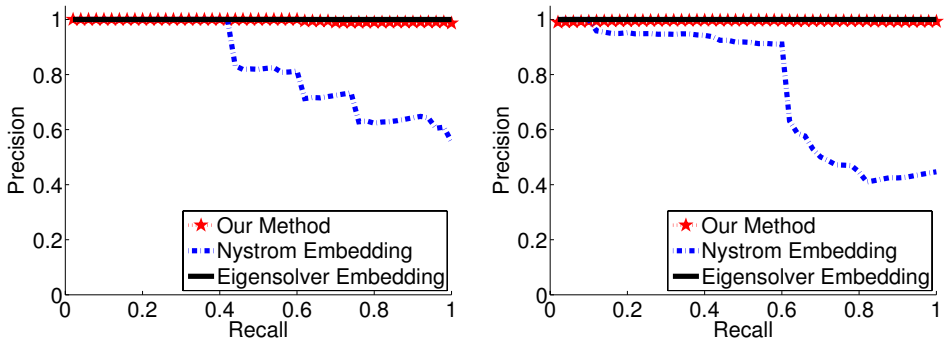
As far as we know, this problem has never been addressed in embedding retrieval literature. Our proposed solution is based on the understanding that the embeddings from *Nyström* and *Eigensolver* are two *distinct* but *highly similar* embeddings. If we can align the query Nyström coordinate to the Eigensolver one, we can carry out the nearest neighbor search by building a spatial indexing tree on Eigensolver embedding. This results in a fast (Nyström) and accurate (Eigensolver) retrieval scheme.

Aligning two arbitrary embeddings and establishing correspondences have been discussed in various literature. The general idea is to exhaustively evaluate each

```
%%% Offline precomputation %%%
A_inv = pinv(A);
% precompute distribution
bar_p1 = sum([A; B'], 1);                          Eqn.5.23
% separate distribution from geometry
A = A./(bar_p1*bar_p1');                           Eqn.5.21
% eigen decomposition
d_Az_si = diag(1./sqrt(sum(A*S,2)));               Eqn.5.11
[V E] = eig(d_Az_si*(A*S)*d_Az_si);
[V,E] = SortEigen(V,E);
% precomputed result
product_VE = d_Az_si*V*pinv(E);
```

```
%%% Online query extension %%%
% append query to landmarks distance to B
hat_B = [B W];                                     Eqn.5.22
% distribution adjustment
p1=sum([A;hat_B'],1);                              Eqn.5.23
p2=sum(hat_B,1)+sum(hat_B',1)*A_inv*hat_B;
bar_p = 1./[p1 p2]';
A = A.*(bar_p(1:n)*bar_p(1:n)');                   Eqn.5.21
hat_B=hat_B.*(bar_p(1:n)*bar_p(n+(1:m))');
% weighted Nystrom
d_Ax = sum([A;hat_B']*S,2);
d_Ax_si = diag(1./sqrt(d_Ax));
d_pi = d_Ax ./ sum(d_Ax);
% extension by precomputation
V_ex=d_Ax_si*([A;hat_B']*S)*product_VE;            Eqn.5.14
% diffusion map embedding
for i=1:size(V_ex,2)                               Eqn.5.4
  V_left(:,i) = V_ex(:,i)./sqrt(d_pi);
end
```

Figure 5.9: Example MATLAB code for computing the Diffusion Map using Distribution Adjustment, Density-Weighted Nyström extension and heuristic precomputation, where $A\{n \times n\}$ and $B\{n \times m\}$ are submatrices of K.



(a) Dog (b) Wolf

Figure 5.10: Retrieval Error due to the Weighted Nyström Extension. Aligning embedding preserves retrieval accuracy.

pair of eigenvectors and find the best correspondence with minimal differences. As pointed out in [105], there are three major problems to align two embeddings: eigenmode switching, sign flip and non-rigid alignment. Eigenmode switching refers to the switching order of eigenvectors when corresponding eigenvalues are close to each other. Sign flip results from the arbitrary determination of signs by eignsolver. Non-rigid alignment refers to the discrepancy between embeddings when they are scaled or skewed.

To handle eigenmode switching and sign flip, we propose to perform a simple search on s leading eigenvectors (Algorithm 2). We set $s = 3d$ for all our experiments. Instead of evaluating Euclidean distance between two whole sets of embeddings, we seek to maximize the value: $\arg\max_{\widehat{\psi_{align}}} \sum_i \psi_i \cdot \widehat{\psi_{align}}_i$, where ψ and $\hat{\psi}$ are the embeddings from Eigensolver and Nyström extension, respectively. $\widehat{\psi_{align}}$ is the aligned embedding of $\hat{\psi}$. The larger the value, the better the alignment is between the two sets of embeddings. Simple dot product is sufficient because ψ and $\hat{\psi}$ are very similar to each other.

After finding the best eigenpairs, we would like to compute the correct query coordinate in the Eigensolver embedding. This can be accomplished by non-rigid alignment using Thin Plate Spline (TPS) [106]. TPS is a coordinate transform method. Given some anchor points, TPS finds the function that passes through the points with minimal error. It then uses interpolation on the function to find the transformed coordinate of an arbitrary point. Since our concern is the query

Algorithm 2: Aligning Eigenmode-Switched and Sign-Flipped embeddings.

Input: $\psi_{1..N}$, $\hat{\psi}_{1..n}$, d, s

Output: $\widehat{\psi_{align}}_{1..d}$

define a set $J \leftarrow 1..s$

for $i \leftarrow 1$ **to** d **do**

  find the eigenpair with largest $\|\psi_i \cdot \hat{\psi}_j\|$ where $j \in J$

  if $\psi_i \cdot \hat{\psi}_j < 0$ **then**

    $\hat{\psi}_j \leftarrow -1 \times \hat{\psi}_j$

  end

  $J \leftarrow J \backslash j$

  $\widehat{\psi_{align}}_i \leftarrow \hat{\psi}_j$

end

coordinate only and a natural correspondence exists between the two embeddings, we apply TPS to the local neighbors of the query $\widehat{\psi_{align}}(q)$, resulting in a fast one-step algorithm. First, we find out the $\bar{n}$ neighbors of the query in the Nyström embedding $\widehat{\psi_{align}}$. Then, we transform the Nyström coordinates of these neighbors into their correspondences in the Eigensolver embedding. The query Eigensolver coordinates can then be obtained by TPS Interpolation. Another note is that since we apply TPS on local neighbors only ($\bar{n} \approx 100$), TPS becomes a small constant cost.

5.4.6 Choosing Landmarks

Since our algorithm is based on Weighted-Nyström, we can generate the set of landmarks using K-mean. We first generate some initial clusters randomly for K-mean clustering. Then, we compute the density and the Nyström embedding, and align it to the Eigensolver embedding. The previously defined dot-product cost is then used for fast pruning of poor embeddings. The higher the score, the better the quality of the Nyström embedding is. We repeat the whole step 15 times simply because K-mean is initialized by random parameters. Once we get the best set of landmarks of a given m, we check if the score is good enough for retrieval. We

compute the Precision and Recall (PR) using such embedding and compare the PR against that of the true embedding. If the error is less than a certain threshold (10^{-4} in our implementation), the algorithm outputs the best set of landmarks; otherwise, we increment m. The algorithm is an offline process.

5.5 Complexity Analysis

After discussing the retrieval framework, we try to give a complexity analysis here. We refer to Figure 5.6 and discuss the complexity in each step.

Offline Process:

1. Computing distance matrix W requires $n\times$ the complexity of the underlying similarity measure, where n is the number of objects in the database. For TPR, it is $O(n_t^3 \log n_t)$ where n_t is the number of topological features. For MRG, it is $O(n_r \times (m_r + n_r))$ where m_r and n_r are the numbers of r-nodes of the two matching models.

2. Parameters optimization step seeks optimal σ and t iteratively. It requires sparse eigensolver. In our implementation, we use Matlab code `eigs`. It is an implementation of Lanczos method which is part of the ARPACK package. However, it is difficult to define a precise running time of such method because it also depends on sparsity which is data dependent. In general, solving eigenproblem for full matrix requires $O(n^3)$. We take this as the worse case of the algorithm. The whole procedure, thus, requires $l \times O(n^3)$ where l is the number of iterations ($l \approx 20$ in all our experiments).

3. To choose landmark objects, we start from a small $m \approx 15$ and increment it by 30. For a given m, we randomly select 15 sets of clusters for the K-mean clustering. We then compute and align the Nyström embedding with eigensolver embedding and compute the alignment cost. It takes around 5-6 iterations to get the best landmark sets. The complexity of alignment step is shown later.

Online Process:

1. Compute query distance to landmarks takes $m\times$ the complexity of the underlying similarity measure, where m is the number of landmarks and $m << n$.

2. Nyström requries $O(m^3)$ for solving eigenproblem and $O(m^2n)$ for extending eigenvector to the whole database. Since the eigenproblem can be precomputed, we only require $O(m^2n)$ to obtain the approximated query coordinate.

3. Alignment step requires $O(s^2) = O(d^2)$ where $s = 3d$ and d is the number of dimensions. We also have to compute Thin Plate Spline (TPS) for non-rigid alignment. However, since we take a fixed number of points in local neighborhood $\bar{n} = 100$, it is a constant cost. The whole alignment step is constant with respect to a predefined dimension d.

The offline process is a slow process with overall complexity $O(n^3)$. It is mainly due to the large eigenproblem that it has to solve. The online process is much faster with overall complexity $m\times$ complexity of similarity measure $+(m^2n)$.

5.6 Experimental Results

In order to test and evaluate the performance of the proposed retrieval framework, we have created a database consisting of 1,020 3D articulating models. They are divided into 13 groups. Some are very distinct (e.g., Frog), while others are very similar to each other (e.g., Dog and Wolf, Lion and Cat) as shown in Figure 5.11. We have incorporated two methods, MRG [17] and TPR [20], into our retrieval framework, and compare the performance in term of accuracy (Section 5.6.1) and speed (Section 5.6.3). We also analyze the proposed alignment step on retrieval performances 5.6.2. All the experiments presented in this section are performed on a PC with a Intel Core 2 Duo 2.33-GHz CPU and 2-Gbyte RAM.

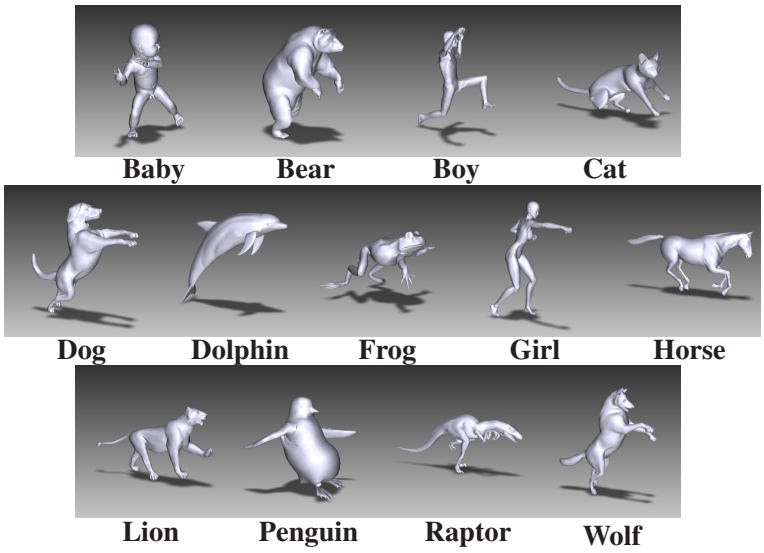
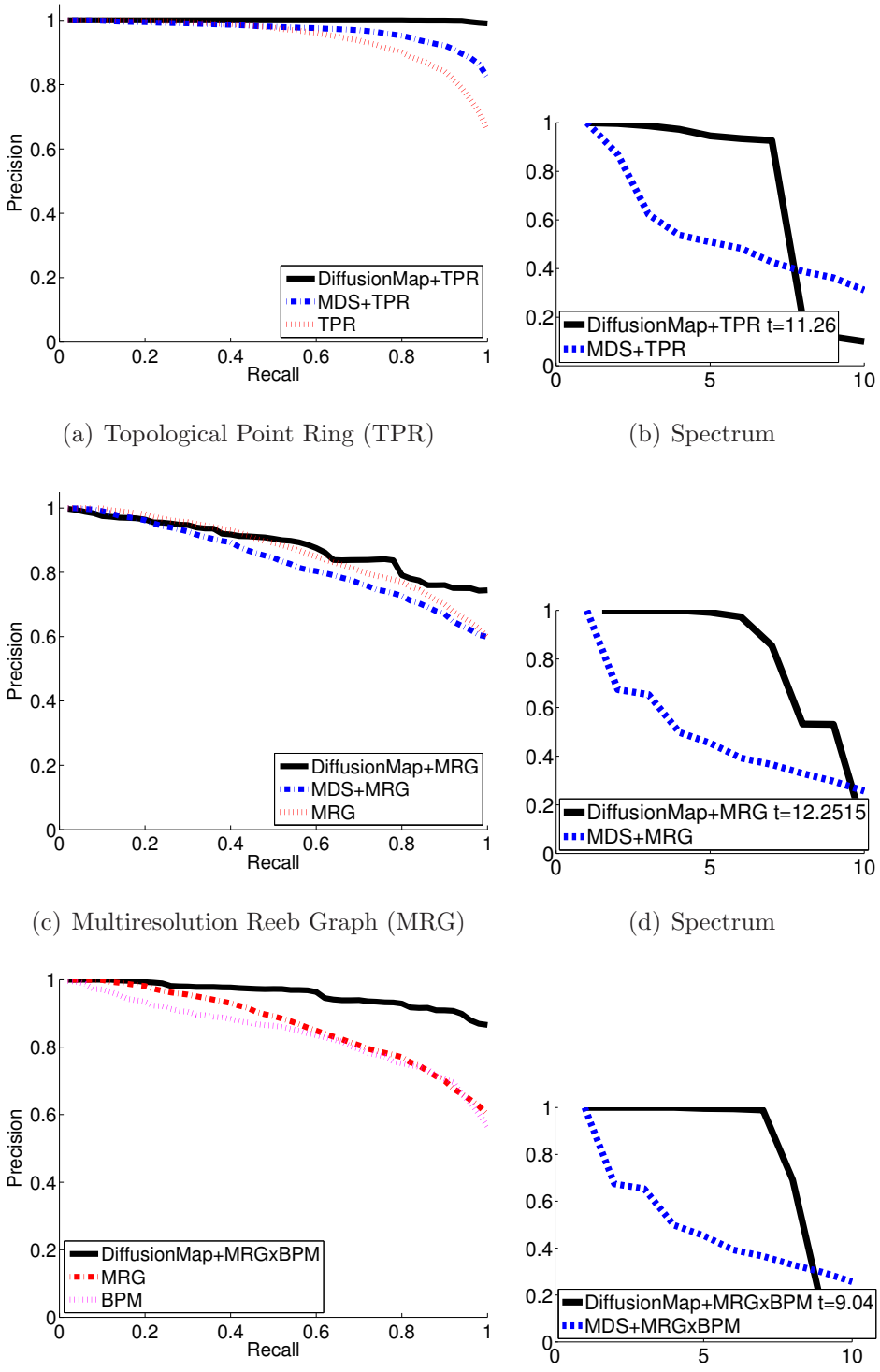


Figure 5.11: 13 groups of models in our database.

5.6.1 Accuracy Comparison

Figure 5.12(a) shows the Precision and Recall (PR) of applying TPR in our retrieval framework. The precision and recall is consistently higher than those of the original method. There is a performance increase of around 30% at high recalls, due to the fact that our framework can effectively maximize inter-class distances among different types of models. The PR curve also shows better performance than Multi-Dimensional Scaling. In MDS, all (far and close) pairwise distances are preserved in the embedding space. However, it is not useful to analyze data lying on manifold. That is why our method can handle these data better.

In Figure 5.12(c), we observe a performance increase when our retrieval framework is applied on MRG [17]. (Note that MDS generally cannot be applied on non-metric distance measure as false dismissal will occur.) The PR curve of MDS is consistently lower than the original method. The performance increase in our method results from the fact that these data lie on manifold. However, the increase is slight. We have diagnosed that the manifold becomes disconnected in some of the model groups, e.g., baby and bear in Figure 5.3 (left). We thus apply the proposed augmented kernel method (Section 5.4.2) using a simple bipartite matching method (BPM) [46]. There is a 20% performance increase at high recall as shown in Figure 5.12(e), and the PR is consistently higher than individual methods. This shows that



(a) Topological Point Ring (TPR)

(b) Spectrum

(c) Multiresolution Reeb Graph (MRG)

(d) Spectrum

(e) MRG augmented by Bipartite Method

(f) Spectrum

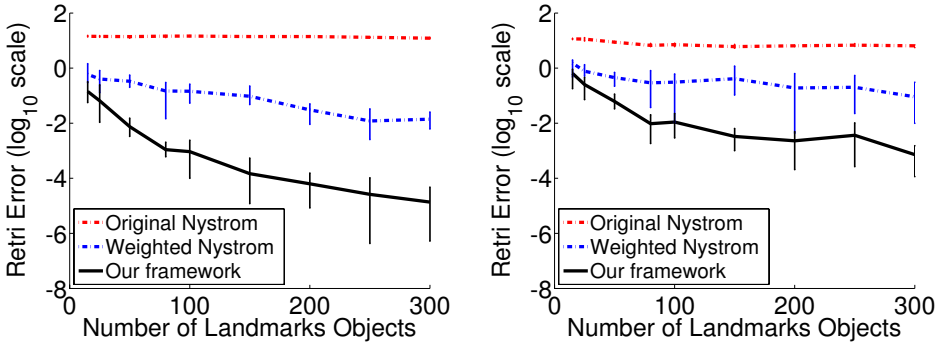
Figure 5.12: PR's of TPR (first row) and MRG (second row), with MDS and diffusion embedding. Third row: PR of augmenting MRG with BPM.

the proposed method can effectively pull these disconnected manifolds together.

For all these precision and recall curves, we also plot the spectrum of MDS and Diffusion Map. Diffusion Map can effectively compress the manifold data in the embedding space into the first few eigenvectors. For MDS, the whole spectrum spreads across 300 eigenvalues. Using merely 10 eigenvectors cannot represent these pairwise distances well. This also explains why our method performs better than MDS.

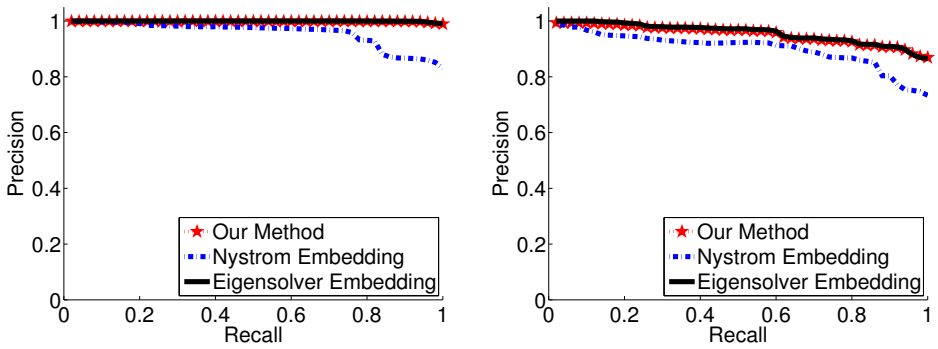
5.6.2 Nyström Alignment and Retrieval Error

As discussed in Section 5.4.5, error due to Nyström extension can greatly impact retrieval performance. Figures 5.13(a) and 5.13(b) (both in logarithmic scale) show the retrieval error due to Nyström extension. By adapting the Density-Weighted Nyström extension, our method performs better than those of original Nyström extension. We make one step further to align Nyström embedding to Eigensolver embedding. Such step preserves retrieval accuracy with retrieval error consistently lower than the two methods mentioned above. Figures 5.14(a) and 5.14(b) also show the PR curves using Nyström embedding with 150 landmarks as well as the PR curves after our proposed alignment step. It should be noted that, we have deliberately remove the query object from the database during the alignment process. The results thus reflect the retrieval effect of a new query. All these results show that our proposed framework is important for the retrieval application.



(a) Retrieval on TPR (b) Retrieval on MRGxBPM

Figure 5.13: Comparing the retrieval error when using Nyström, Density-Weighted Nyström and our approach.



(a) Precision/Recall on TPR (b) PR on MRGxBPM

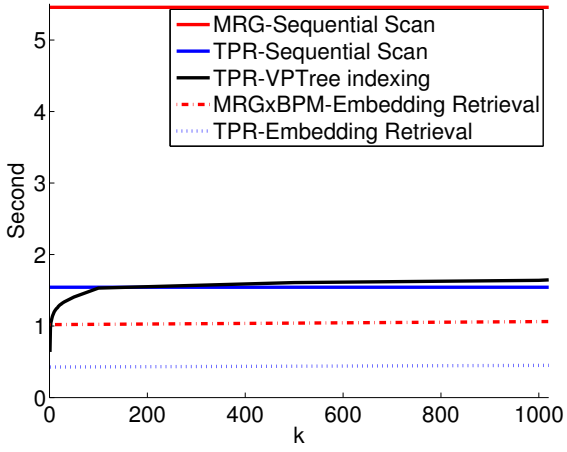
Figure 5.14: Comparing the Precision and Recall when using Eigensolver embedding, Density-Weighted Nyström embedding and our approach.

5.6.3 Time Comparison

Figure 5.15 compares the retrieval time per query. Since MRG (a graph-based method) is non-metric, indexing techniques cannot be applied. The retrieval time for a query is long as it involves sequentially scanning the whole database. TPR (a bag-based method) is metric, and distance-based indexing (VP-tree) is applicable. However, due to the high intrinsic dimension of geometric features, the method soon approaches brute-force. Our embedding retrieval framework can incorporate both methods whether metric or non-metric. Since our framework only requires comparing the query with a small set of landmark objects (150 objects, about 15% of the whole database), it is roughly 3 (TPR) and 5 (MRG) times faster than their corresponding sequential searches. Since a spatial indexing technique (Kd-tree) is extremely fast, the time spent on k-nn search is so small that it can be neglected compared to the time spent on evaluating the similarity measure. Hence, when our retrieval framework is applied on the two methods (MRG and TPR), the computational costs are roughly constant as the required number of nearest neighbors increases. In addition, our method does not suffer from the curse of dimensionality because of the reduced dimension. Further, our method consistently performs faster than the indexing approach even when k is small.

In our experiments, the time for comparing two models using TPR and MRG are

based on C++ implementation (Section 4.7). For all the other discussion and experiments, we use Matlab 2007 for implementation. These include Diffusion Map, the kernel method, general and Density-Weighted Nyström extension and the proposed Nyström alignment scheme.



knn	TPR (s)			MRG (s)	
	TPR	VpTree	DM	MRG	DM
1	1.54	0.638	0.442	5.46	1.055
5		1.11	0.442		1.055
10		1.202	0.442		1.055
20		1.287	0.443		1.056
30		1.337	0.442		1.055
50		1.406	0.443		1.056
100		1.531	0.443		1.056
500		1.609	0.446		1.059
1000		1.639	0.45		1.062
1020		1.646	0.45		1.063

Figure 5.15: Retrieval Time Comparison of sequential scan (MRG, TPR), distance-based indexing (TPR+VPTREE), and our framework (MRG, TPR).

5.7 Conclusion

In this chapter, we have pointed out that MRG (graph) and TPR (bag-based matching) methods cannot handle 3D articulated geometry models well with similar skeleton, as they project data on nearby manifolds. Increasing the number of geometric features may increase the distance between these manifolds but the intra-class variance is usually larger than the inter-class distance. Retrievals using these methods may fail especially when the database contains many similar skeleton models. Our idea here is to use manifold learning algorithms to maximize inter-class distance. To handle large databases, we have also adapted the Density-Weighted Nyström extension for the computation of the Diffusion Map and use correspondence analysis to define a retrieval framework to reduce Nyström extension error. We have shown with a number of experiments that the proposed framework improves retrieval accuracy and speed significantly.

Chapter 6

Feature Extraction and Matching for 3D Motion Captured Data

6.1 Introduction

Apart from 3D articulated geometry models, 3D motions are also important to nowadays graphics application. It is used not only in animated films, but also in computer games and crowd simulations. There are a number of methods developed to produce human motion data. A well-known method is called motion capture (MoCap). In this method, motion sensors are attached to various parts of a human actor. As the human actor performs some specific movement, the computer records the position of each sensor at each time frame. This creates a series of motion frames, known as time-series. With this method, real human motions can be captured and used to drive the movement of virtual characters (e.g., 3D articulated geometry models). By joining several time-series together, more complex human motions can be created [2], [3]. However, as the number of motions in the database grows, it becomes difficult to select an appropriate motion that satisfies certain requirements. As such, motion retrieval has become one of the major research focuses in motion capture animation in recent years.

To design a reliable motion retrieval method, there are many challenges to address. First, a human model contains many joints and human animation consists of many time series, each representing the motion of a single joint. A method should

be fast enough to enable interactive retrieval of matched motions despite the size of the database. Second, similar motions may still be differed in many ways, such as length, local shifting, local and global scaling, which affect the matching accuracy. While most existing methods use Dynamic Time Warping or Uniform Scaling, we consider another method to handle the matching problem here.

We have first observed that motion sequences may consist of local shifting, local and global scaling and they are rather similar in shape. We have also observed that existing matching methods suffer from restricted matching where frames are matched to nearby frames in a temporal manner only. This limits the improvement on accuracy and speed. To match these sequences and improve efficiency, we propose to model the temporal matching problem as a bag-based matching problem. By using Earth Mover's Distance, an efficient and accurate method can be defined. However, because EMD allows morphing from any frame of a motion to any frame of another motion, strayed matching may result. To solve this issue, we propose a ground distance that penalizes strayed matching. Our experimental results show that the proposed method is promising.

Since this chapter discusses an independent application, we present the chapter as a coherent whole. In Section 6.2, we summarizes existing motion matching and retrieval methods. Section 6.3 presents our method in detail. Section 6.4 evaluates the performance of our method through a number of experiments. Section 6.5 evaluates the experimental results and Section 6.6 briefly concludes this work.

6.2 Related Work

Motion retrieval research is still relatively new compared to retrieval research of other multimedia data. There are only a few motion retrieval methods in the literature. A number of them are extended from or strongly related to existing audio retrieval methods. A good example is the Dynamic Time Warping (DTW) method. DTW is a signal processing technique used to find a non-linear alignment between two signals, while minimizing the error between them. Many audio and speech processing algorithms as well as music retrieval methods apply DTW extensively. As

motion data can be considered as multi-attribute time-series, applying DTW on motion retrieval is straightforward. Many motion retrieval systems use DTW as the similarity measure [2], [107], [108], [109], [110].

However, DTW has low efficiency. In addition, as motion capture data consists of many parameters and attributes, dimension reduction methods are typically employed to reduce computational complexity. The GEMINI framework, proposed by [111], was one of the earliest examples. It first extracts a low-dimensional approximation for each time-series in the database. Then a distance metric is defined to lower bound the approximation. Usually these low-dimensional approximations are stored in a spatial data structure, like R-Tree, for fast retrieval. There are many dimension reduction methods, which all adapt similar framework. Notable examples include Fourier transform [112], wavelet transform [113], average values in adjacent windows [114] and bounding boxes [82]. In order for DTW to support indexing, [8] proposes the bounding envelopes, similar to the GEMINI framework. [9] further suggests that similar motions which are differed by uniform scaling cannot be matched by DTW because DTW can only handle subtle local differences after the best globally scaled match has been found [115]. Hence, [9] proposes an algorithm which is based on uniform scaling to match the query to those globally scaled candidate motions. However, based on our understanding, in order to handle motions that contain both local and global differences, one needs to apply DTW and Uniform Scaling separately, significantly increasing the computational cost.

Recently, a geometry based method is proposed [116] by extracting boolean features from geometric relationships of motion data. These qualitative features are compact and descriptive. [117] extends the work by introducing a linear-space indexing structure, fuzzy queries and adaptive fuzzy hits. Although these features are descriptive, they require textual descriptions to be used as queries, and users need to define specific geometric features for comparison. Subsequent work [118] considers the method for the creation of motion templates. [119] provides a GUI that eliminates the need for textual description. On the other hand, [120] proposes to segment and cluster geometric features automatically into an indexing tree, and a matching algorithm based on peak points. However, in general, these methods

concern finding logically similar motions but not a close match.

Our method also considers dimensionality reduction. However, we extract features based on the Douglas-Peucker algorithm, which was originally proposed as a line simplification method to reduce the number of points in data representation [121]. Our method differs from all previous methods in that we do not necessarily require a fixed number of dimensions. This has the advantage that our method can best preserve the original motion. This is particularly good for joints such as hands, where vigorous motions are expected and we may use more features to represent them. On the other hand, for those stationary joints such as head, our method extracts only a few features and thus reduces computational cost.

In this part of the research, we are interested in finding motions that are entirely similar to a given query. While a typical motion retrieval method may only handle local shifting, local scaling or global scaling, we try to tackle the same problem from another point of view. We convert the matching into a transportation problem. We compute the minimum energy to morph one motion sequence to another using EMD. As this work focuses on accuracy analysis, we compare mainly with DTW and Uniform Scaling method. Though we have not implemented any indexing scheme, extending our method to support indexing can be easily achieved because our distance function is a metric.

6.3 Method Overview

Suppose that we want to determine the similarity between two motion sequences,

$$Q = q1, q2, \cdots, qN$$

$$C = c1, c2, \cdots, cM$$

where Q is the query motion of length N and C is the candidate motion of length M. It is important that the similarity method considers each feature of Q and C when comparing the data, where the features are extracted from Q (or C) and are used to represent Q (or C). Obviously, different similarity methods use different features for comparison. For example, Euclidean distance compares the distance of

the joint position (or angle) in each frame of Q and C. The number of features is therefore equal to the number of frames. This implies a very high dimensionality of the feature space. Hence, it is favorable to reduce the dimensionality of this space and therefore the computational complexity. On the other hand, we also need to minimize the data loss during this dimensionality-reduction process. As we reduce the dimensionality of the feature space, we must exploit properties of the data to facilitate this procedure.

We propose a new approach to solving the human motion retrieval problem. To reduce the dimensionality of the feature space, we apply the Douglas-Peucker curve trimming algorithm. This reduction technique gives a very tight approximation of the time-series data for each joint rotation. By considering Douglas-Peucker points as pieces of mass and holes in space, we can effectively apply the EMD method to the point sets. The EMD value returned provides a similarity score between the query and candidate. Applying this technique to every joint of the human skeleton, the summation of the values provides a score for the overall skeleton animation. In our implementation, we weight the joint scores according to their perceptual importance. For example, arm rotations receive a higher weighting than finger rotations since arm rotations are perceptually more important in general.

6.3.1 Feature Extraction

In reducing the dimensionality of the features (i.e., frames) from N in n features, $n < N$, we apply the Douglas-Peucker (D-P) algorithm to the motion data. In human motion data, the joint rotation value from one frame is generally similar to its predecessor and successor frames. This property of motion data allows the application of D-P approximation.

6.3.1.1 The D-P Algorithm

Commonly used in data compression, the D-P algorithm is a recursive procedure that represents time-series data using a series of straight lines. Given a time-series Q and a value ε, the D-P algorithm begins with a single line with end points at $q1$ and qN. For each frame in Q, D-P computes the distance from the line, storing the

maximum distance of any point from a line. If the maximum distance is greater than ε, from the line, the line is split into two lines and the D-P algorithm is then called on the two line segments produced. In this manner, the algorithm recursively splits the line until there are no frames greater than ε from some line. Figure 6.1 illustrates this operation.

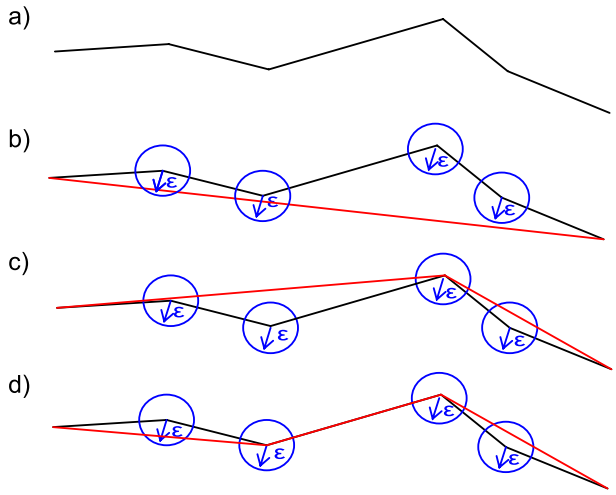


Figure 6.1: Recusive Douglas-Peucker Algorithm

6.3.1.2 Feature Representation

To apply the D-P algorithm here, each line segment representing the motion of a joint in time contains a start point and an end point. Each point (or feature in our method) contains x, y and z rotation values, along with a t value indicating the time position of that point, i.e., the frame number, and an m value indicating the number of frames it encapsulates (or the mass as will be explained in Section 6.3.2). We define the i-th feature as follows:

$$< x_i, y_i, z_i, t_i, m_i >$$

Hence the feature set of one joint motion becomes:

$$F = \{< x_1, y_1, z_1, t_1, m_1 >, \cdots, < x_n, y_n, z_n, t_n, m_n >\}$$

Note that the D-P algorithm is a lossy compression algorithm and therefore some data loss is unavoidable. The accuracy of the approximated data depends on the number of features used to approximate it, and thus the threshold ε.

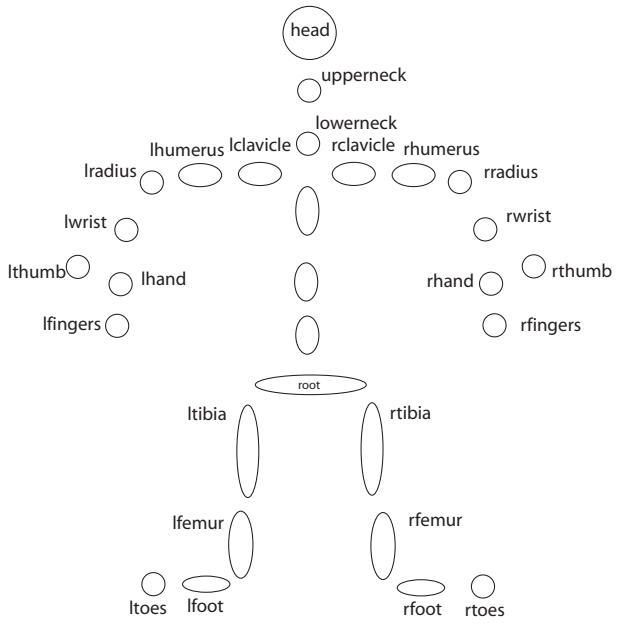


Figure 6.2: ASF Motion Hierarchy

6.3.2 Feature Matching

For feature matching, we apply the EMD method on the features extracted by the D-P algorithm. To our knowledge, the EMD method has never been applied to human motion matching before, although it has been applied to image retrieval [86] as well as model retrieval (Section 4) with considerable success. By applying the EMD to the extracted features of each joint in the human body, the summation of results for each joint provides an overall dissimilarity score for the entire skeleton. In Figure 6.2, we gives an illustration of various joints which are detailed in ASF (motion and hierarchy) file format. As mentioned earlier, we weight each joint according to its importance manually. In general, we assign smaller(larger) weights to joints that correspond to smaller(larger) components. These weights are detailed in Figure 6.3.

Given the extracted features of the query, F_q, and those of the candidate, F_c, the dissimilarity score is computed as follows:

$$D_{EMD}(F_q, F_c) = \sum_{j=1}^{|J|} W_j \times EMD(F_{q,j}, F_{c,j}) \tag{6.1}$$

where W_j is the weight of joint $j \in J$.

```
root = 100
lowerback = 20
upperback = 20
lowerneck = 10
upperneck = 10
head = 5
thorax = 20
lclavicle = 70
rclavicle = 70
lhumerus = 60
rhumerus = 60
lradius = 50
rradius = 50
lwrist = 7
rwrist = 7
lhand = 3
rhand = 3
lfingers = 1
rfingers = 1
lthumb = 1
rthumb = 1
lfemur = 70
rfemur = 70
ltibia = 60
rtibia = 60
lfoot = 7
rfoot = 7
ltoes = 1
rtoes = 1
```

Figure 6.3: Weights of Joints.

6.3.2.1 Feature Matching and EMD

In the EMD framework (see Section 4.6.1 for detail), we are allowed to specify the *mass* of each feature. It is a value that describes how important the feature point is. The larger this value, the higher the importance. To explain it in physical terms, we can indicate the mass of a feature as the size of a circle as shown in Figure 4.16. In our implementation, the mass is stored as one of the components in a feature, $F[m_i]$, as described in Section 6.3.1.2.

To compute the ground distance, d_{ik}, to transport one unit of mass from one location to another, we consider the Euclidean distance between each dimension of a feature point, x, y, z and t as follows:

$$d_{ik} = \sqrt{(F_q[x_i] - F_c[x_k])^2 + (F_q[y_i] - F_c[y_k])^2 + (F_q[z_i] - F_c[z_k])^2 + (F_q[t_i] - F_c[t_k])^2} \tag{6.2}$$

where F_q and F_c are the features extracted by the D-P algorithm from the query and the candidate, respectively, for one joint. In the 3D motion clips, x, y and z

represent the joint angles in degrees. In our implementation, t is the unit of time frame, which is supplied by the motion clip, and we apply no normalization in our calculation.

Using the constructed cost matrix, the optimization of the flow matrix may help determine the similarity between the query and the candidate. For example, if a query contains a section where a subject raises his hand and then lowers it, and the candidate also contains similar motion, then there would be a high flow between these features requiring a low amount of energy. On the contrary, if the candidate does not contain this motion, the features of the query must be transported to some feature(s) of the candidate using a high amount of energy. This results in a higher overall energy transfer, indicating a poor similarity between the two motions.

6.3.2.2 Comparison to the Sakoe-Chiba Band

In our method, we penalize the transfer of a piece of mass to a hole with a different start frame t. As pointed out by Mr Mark Corbyn, we observed that this produces a similar effect as the Sakoe-Chiba band, which is widely used in Dynamic Time Warping.

There are two important properties in applying the Sakoe-Chiba band in DTW. First, it prevents frame matching from straying too far from the diagonal of the matrix, i.e., matching frames which lie outside the Sakoe-Chiba band. Second, it can provide a significant speed improvement. This is achieved by restricting the frame matching computation process to only those frames that lie within the Sakoe-Chiba band [122].

By considering the t value in our method we restrict frame matching to within the diagonal region. This is similar to the first property of the Sakoe-Chiba band. However, while the Sakoe-Chiba band implements a sharp cut-off point beyond which frames cannot be matched, the weighting in our Ground Distance provides a gradual restriction (Figure 6.4). As a result, our method does not have the speed improvement as the Sakoe-Chiba band does because our method needs to compute the entire cost matrix.

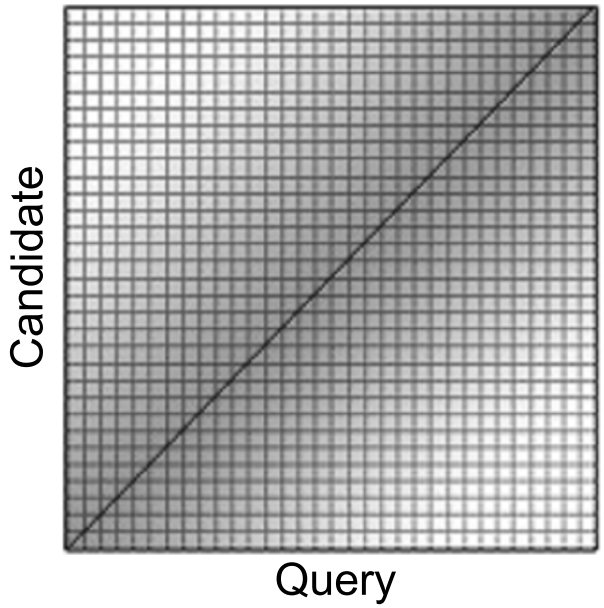


Figure 6.4: Penalizing strayed matching from the diagonal and its similarity to Sakoe-Chiba band.

6.4 Experiments and Results

To evaluate the performance of the proposed matching method on different motion clips, we discuss some of the experiments that we have conducted. We have constructed a small motion database from 115 different motion clips. We categorize the 115 motions into 5 motion groups, climbing, jumping, running, sword playing and walking. All the experiments presented here are performed on a PC with a Pentium 4 3GHz CPU and 1GB RAM. All experiments presented here are based on our Java implementation. The motion files are downloaded from CMU [123].

The motion clips typically contain more than one action within each clip. To obtain more accurate performance results, we manually break each of the clips down into basic motion clips with a single action. The basic motion clips have different lengths varying from 150 to 600 frames. We down-sample all the motion clips into 128 frames while ensuring that the down sampled motion clips do not have apparent artifacts. In our experiments, DTW and our method use only these down-sampled motion clips as input. However, in order to allow scaling in Uniform Scaling, we use the basic motion clips as input and we take the first 128 frames of the basic motion clips as the query for scale computation. This is due to the fact that down

sampling may produce the effect of scaling. Our objective in the experiments is to find the most similar motion clips within the motion database. For comparison, we have implemented Dynamic Time Warping and the Uniform Scaling method. In our DTW implementation, we use a Sakoe-Chiba Band of width r=10% of the basic motion to control local shifting and scaling. In Uniform Scaling, we use a maximum scaling factor of 1.2 to search for the best suitable scale. For time comparison, we also apply bounding envelopes for fast pruning of irrelevant motions.

6.4.1 Performance on Motion Discrimination

In the first experiment, we compare the retrieval performance of the three methods, DTW, Uniform Scaling and our method, using the similarity matrix. To generate the matrix, we first compute the similarity score between every motion pair in our database. We then normalize the results with the maximum and minimum of the corresponding matrices to show the contrasts. The darker the color, the more similar the two motions are.

Figure 6.5 shows the similarity matrices of the three methods. To improve readability, we have clustered the similarity results according to the five motion groups. They are labeled as well as highlighted with five different colored squares in the diagrams. These similarity matrices, M, are also normalized into the same range [0,1] to compare their contrasts, using the following formula:

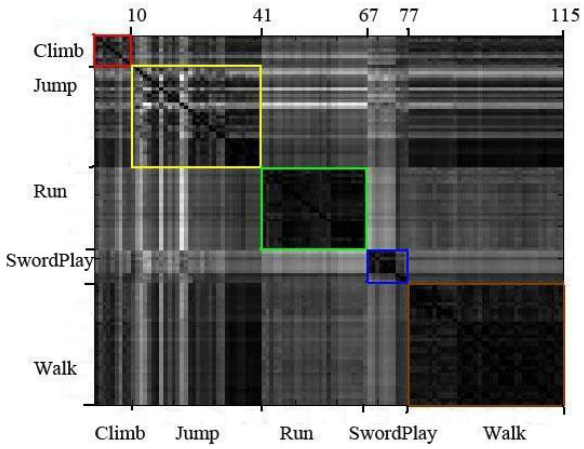
$$M(x,y) = \frac{M(x,y) - \min(M(x,y))}{\max(M(x,y)) - \min(M(x,y))} \tag{6.3}$$

From the similarity matrices, we have the following observations:

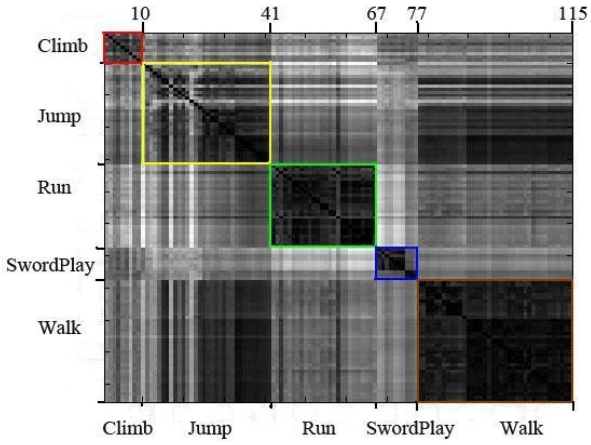
1. The diagonal lines of the three matrices give the darkest color. This means all three methods perform well in identifying the same motion.

2. In general, the results within the five colored squares show dark color for the three matrices. This means that all three methods are able to give high similarity scores for similar motions.

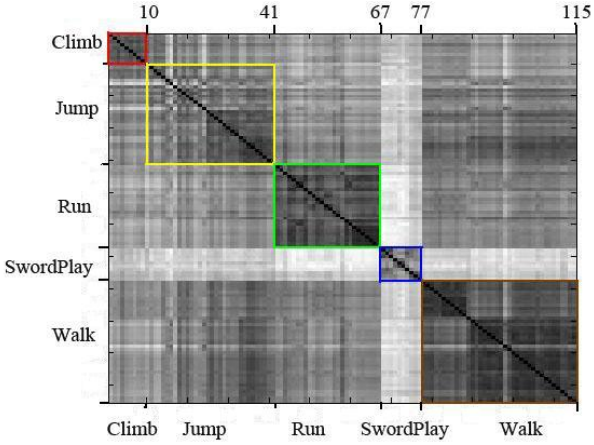
3. Our method also gives a larger similarity contrast than DTW and Uniform Scaling when comparing two motions from different groups, as the similarity



(a) Uniform Scaling



(b) Dynamic Time Warping



(c) Our Method

Figure 6.5: Similarity Matrix of Uniform Scaling, DTW and EMD

matrix of our method generally gives lighter colors outside the five square regions. This may suggest that our method can distinguish different motion groups relatively easier with high contrast.

6.4.2 Performance on Motion Retrieval

In the second experiment, we compare the retrieval performance of the three methods using the average precision and recall graph (Figure 6.6). This graph is generated by taking each of the motions in the database as query, searching similar motions from the same database and averaging all the precision and recall values.

Figure 6.6 shows our precision and recall results. From the diagram, our method performs much better than the other two methods because our method always gives a better precision for any given recall, especially when the recall is above 0.1. This finding confirms our similarity analysis that our method can distinguish dissimilar motions. Figure 6.7 plots the precision and recall for each of the motions in our database.

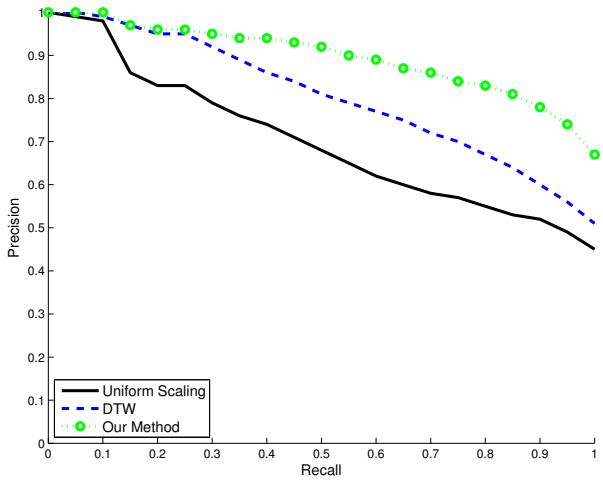


Figure 6.6: Average Precision Recall.

From Figure 6.7, we can see that all three methods perform very well in the run motion, whilst our method performs better in the climb, jump, sword play and walk motions. All three methods have similar good performance in the run motion may suggest that all these run motions are highly similar to each other within the group and are very distinctive from motions of other groups. This also accounts for the

high precision (≥ 0.9) at high recall (1.0). Currently we can only perform precision and recall analysis on five groups of motion data because most data available at [123] consist of many different kinds of motions and cannot be manually classified into a single meaningful group.

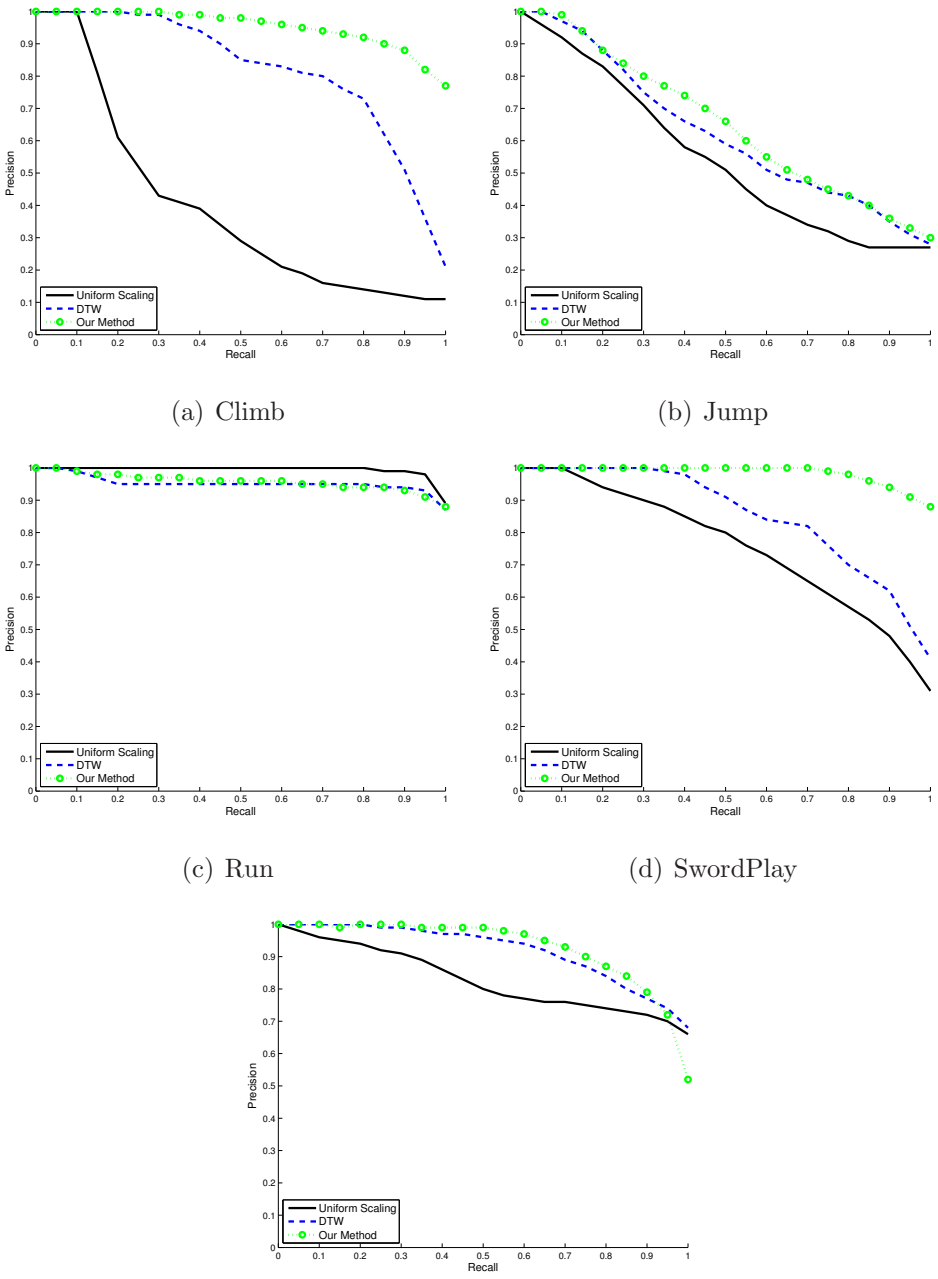
6.4.3 Speed Comparison and Complexity Analysis

In the third experiment, we would like to compare the performance of the three methods based on speed and computational complexity. In Uniform Scaling, we try to find the best scaled match between the query and the candidate. So, the time complexity is $O(p \times (M - N))$, where p, M and N represent the lengths of a scaled time series, the candidate and the query, respectively. The time complexity of DTW is roughly $O(MN)$.

The time complexity of EMD used in our method is harder to analyze because it is based on the simplex algorithm. However, according to our earlier analysis (Section 4.7.1.4), if the algorithm is modeled as a bipartite matching problem, the complexity is $O(n^3 \log n)$, where n is the number of features. Based on this time complexity, it may appear that DTW would perform better than EMD method. However, our experimental results as shown in Table 6.1 reveal that EMD actually performs better than DTW. This is because DTW computes all the motion frames (128 frames) but EMD, by applying the D-P algorithm, involves a far smaller number of features. From our experiments, the number of features used in the EMD algorithm varies from 2 to 30. This explains why the computation time consumed by EMD is far less than DTW method.

	Uniform Scaling	DTW	EMD
Climb(10 files)	11.5 s	9.8s	6.0s
Jump (31 files)	104.4 s	124.9s	12.8s
Run (26 files)	242.9 s	230.8s	11.8s
SwordPlay (10 files)	49.3 s	59.2s	2.0s
Walk (38 files)	220.8 s	359.0s	18.3s

Table 6.1: Feature Matching Time of Uniform Scaling, DTW and EMD



(a) Climb

(b) Jump

(c) Run

(d) SwordPlay

(e) Walk

Figure 6.7: Precision and Recall of Uniform Scaling, DTW and EMD

To explain why n varies so much in the EMD method, we may consider human movement. Different parts of the human body have different ranges of movement when performing different actions; some joints may vary a lot during a particular action while others may move very little. For example, during walking, most of the time the human's head will not change much while the feet may move continuously. This causes n to vary from joint to joint and from motion type to motion type.

To extract the feature points with the D-P algorithm, we need to define a threshold ε (Section 6.3.1.1) as a condition to determine if a line should be subdivided. If this threshold is set to be too large, the number of features produced may be too small. On the other hand, if it is set to be too small, then the number of features produced may be too large, with a lot of unnecessary feature points. In our experiment, we determine ε empirically and set it to 10. This applies to all motion clips in our database.

6.5 Discussion

From our experimental results, we can see that our method performs relatively better than DTW and Uniform Scaling in terms of accuracy and speed. There may be two reasons that can explain such performance improvements. First, DTW can only handle local scaling and shifting while the Uniform Scaling method can only handle global scaling during the motion matching process. On the other hand, our method, as suggested by the experimental results, may be able to handle local shifting, local and global scaling simultaneously because our method depends on an energy morphing technique, whilst DTW and Uniform Scaling are designed to handle local shifting and global scaling respectively.

Second, our database contains motion clips of significantly different lengths, i.e., different numbers of frames. As we use the settings defined in the original experiments in the corresponding papers (e.g., DTW alignment range r=10% and Uniform Scaling maximum scaling factor = 1.2), they may not find a best match in our motion data.

6.6 Conclusion

In conclusion, we have introduced a novel and efficient method for retrieving human motion data. Unlike other approaches, our method applies the Douglas-Peucker (D-P) algorithm to extract features according to the movement of the motion. As the number of features is not fixed, our method can preserve more vigorous motion by using more features. Whilst for relatively stationary joints, it reduces the number of features to reduce the computation time. To analyze the similarity between two motions, we model it as a transportation problem, and apply the Earth Mover's Distance (EMD) method as the matching framework. Our experiments show encouraging results as compared to the Dynamic Time Warping and Uniform Scaling methods.

Chapter 7

Evaluation

In the previous chapters, we have presented our proposed solutions and give experiments for comparison. In this chapter, we would like to evaluate the whole research study. We first give a more complete view of our method by comparing it with existing works. Then we discuss the strength and weakness of our methods.

7.1 Performance Comparison with BDLA

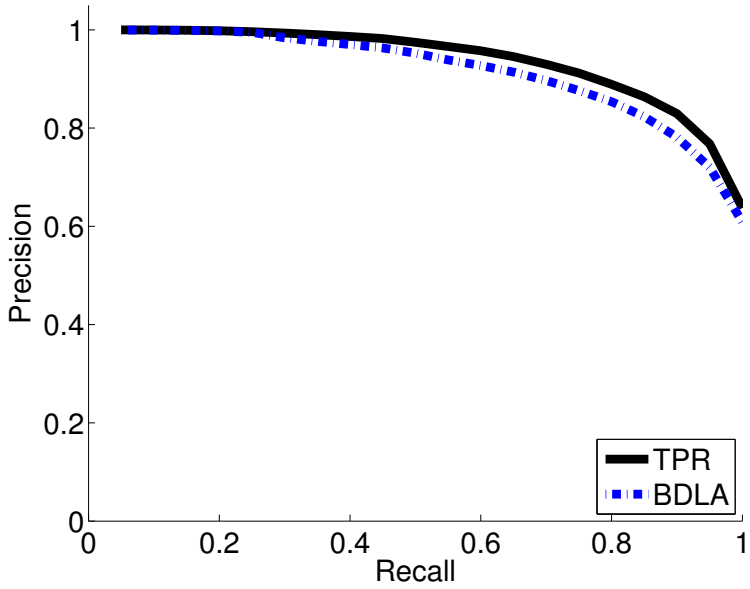


Figure 7.1: Precision and Recall of TPR and BDLA on 902 models.

Recalled that our method, Topological Point Ring (TPR) Analysis, is a sequel

of our previous work [19]. We try to compare them here. In our previous work, we proposed a Bi-Directional LSD Analysis (BDLA). The idea is to capture topological bounded regions as features. These bounded regions are extracted solely based on critical points. It performs well on simple articulated geometry models. However, as pointed out in Section 4.3.4, such method may still be affected by noise and produce extraneous critical points, resulting in a lot of small features. In Section 4.5.1, we have proposed a voting method to choose reliable critical points as topological points. It solves the extraneous critical points and slicing direction problems. In Section 4.5.3, we have also proposed a multi-source point method to capture topological rings.

We have applied the two methods in our model database. When BDLA was applied on more complex models, it crashed by excessive noise. Among 1020 models, only 902 models are successfully processed. On the other hand, TPR reliably handled all the models without problem. It should be noted that BDLA does not provide a metric similarity measure as there are overlapping areas between different bounded regions. Thus, indexing techniques cannot be applied. In TPR, the similarity measure is ensured to be metric, and indexing techniques are applicable. We also compared the precision and recall of the two methods in Figure 7.1. TPR performs relatively better than BDLA on these 902 models. This shows that the new feature representation is reliable and more stable to represent 3D articulated geometry models.

7.2 Performance Discussion of TPR

In Chapter 4, our proposed feature extraction method (TPR) is tested on a small database (150 models) featuring rotation and scaling. Here, we try to compare the performance of TPR with MRG using a larger database (1020 models) as mentioned in Chapter 5. We compare the overall Precision and Recall curves in Figure 7.2.

From the figure, we see that TPR and MRG outperform D2 and Fourier methods, where both are non-articulated matching methods. The performance as represented by the precision and recall also confirms that both TPR and MRG are indeed meth-

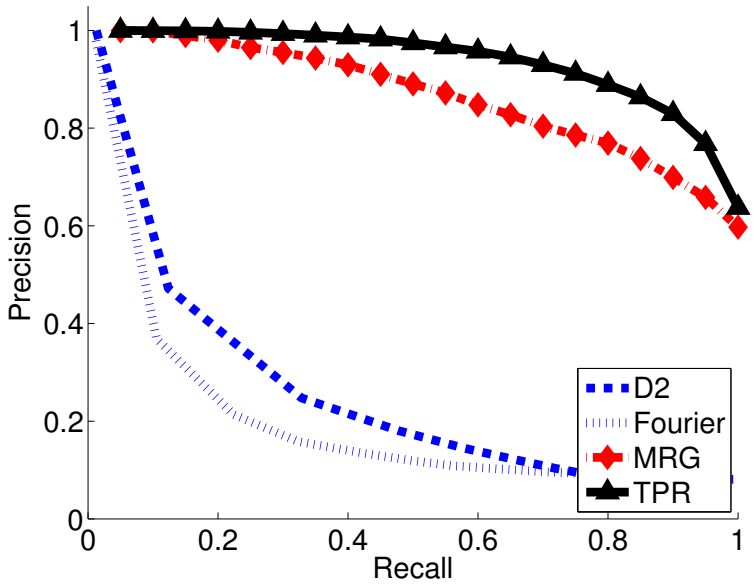


Figure 7.2: Precision and Recall of D2, Fourier, MRG and TPR method on 1020 models.

ods that can handle articulated geometry models. In addition, it confirms the results in Chapter 4 that TPR outperforms MRG.

To further evaluate the performance of TPR, we have plotted the individual precision recall curves in Figure 7.3. There are two observations.

1. TPR performs similarly or even better than MRG in groups baby, bear, boy, cat, dog, dolphin, horse, penguin, raptor and wolf; but performs poorer in groups girl and lion.

2. There is a sudden drop in precision of both methods (TPR and MRG) at high recall. e.g. bear, boy, cat, girl, horse, lion, penguin and wolf.

To explain these observations, we review the findings in Chapter 5. TPR captures geometric histogram based on geodesic. MRG partitions a mesh using integral geodesic. As explained earlier (Section 5.2), when comparing two highly similar skeleton models, graph and bag-based matching become Euclidean distance (Figure 5.4). However, Euclidean distance is not a smooth function with respect to the natural parameters (deformation in our concern). Though it is generally believed that geodesic and integral geodesic is stable towards articulation, a small misalign-

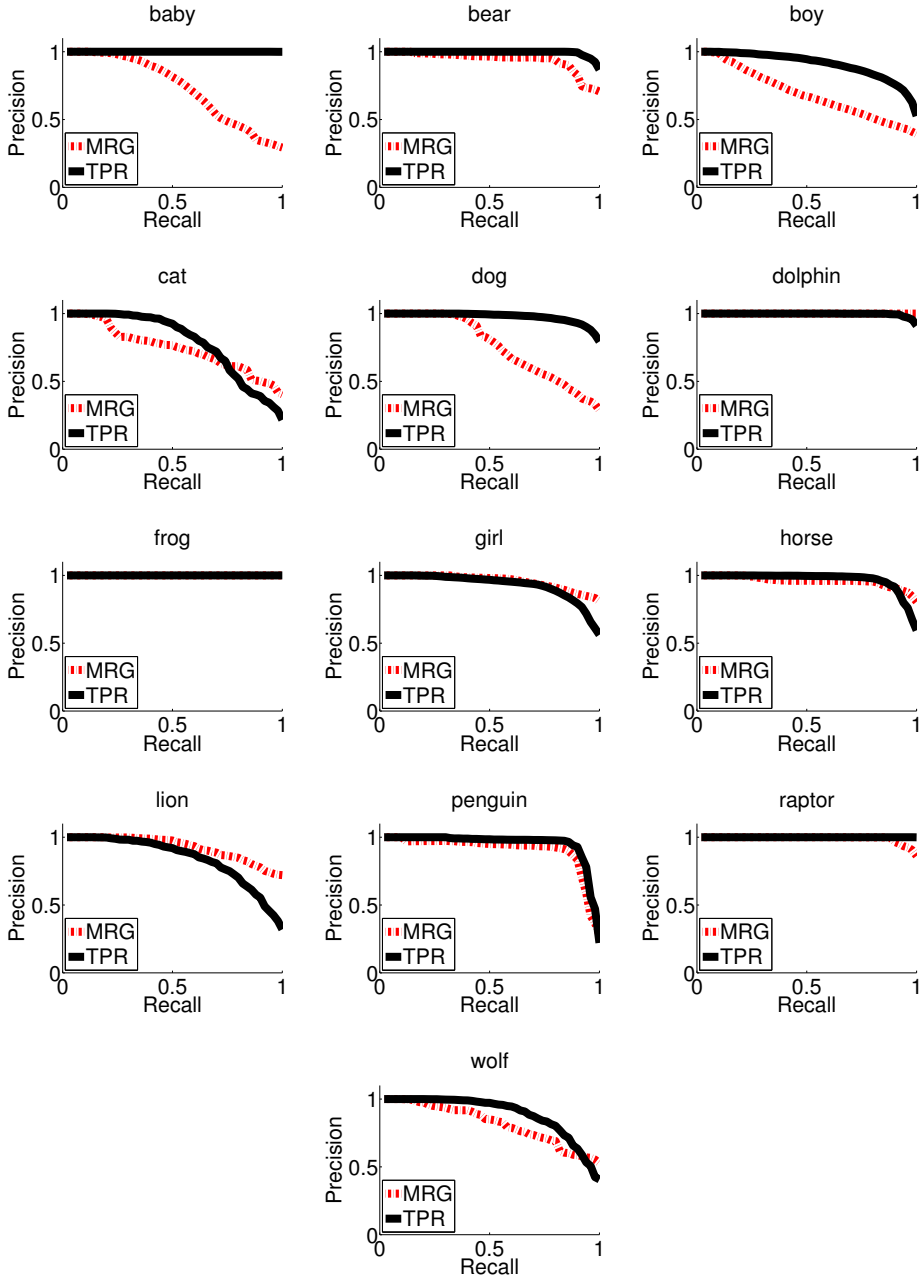


Figure 7.3: Comparing TPR and MRG on individual Precision and Recall.

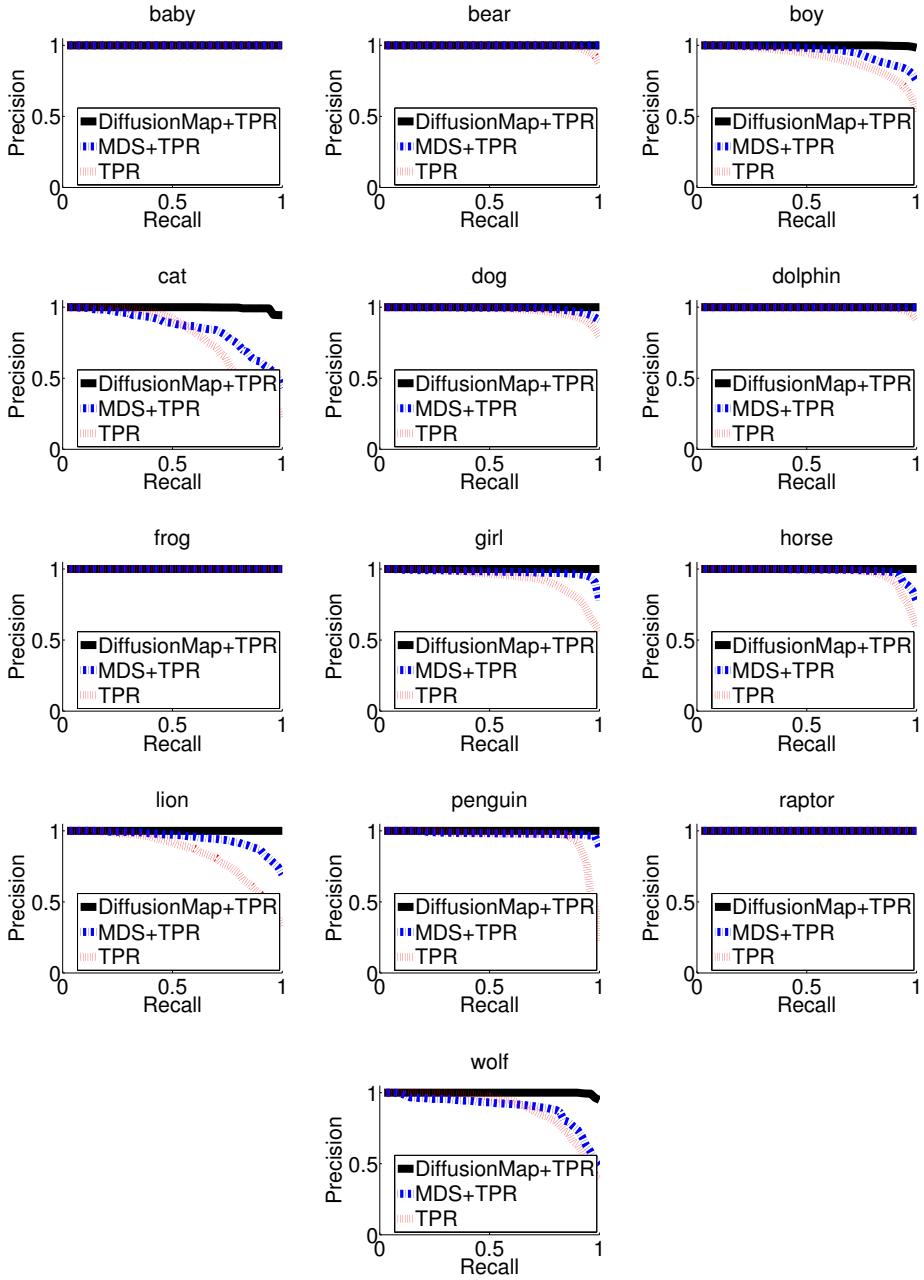


Figure 7.4: Comparing the individual Precision and Recall before and after the application of manifold learning techniques.

ment would lead to large group variance in the resulting similarity measure. Such situation would even be amplified due to quantization. All these explain the sudden drops in the precision of both methods at high recall.

The reason that TPR performs poorly in girl and lion compared to MRG can also be explained by this fact. The performance drops when the variance is larger than inter-class distance. Recall that the database contains many highly similar skeleton models (e.g., girl - boy and baby, lion - cat, dog, wolf and horse). This is evidenced by Figure 7.4. When we apply our embedding approach to TPR, all the groups are well separated as shown in the corresponding precision and recall. This suggests that the features captured by TPR suffer from geodesic misalignment problem.

Figure 7.4 also confirms that the proposed embedding retrieval framework (Chapter 5) can effectively handle manifold data with large variance due to misalignment. As seen in the figure, all the precision and recall of individual group performs well. Many of the groups attain 100% precision at 1.0 recall. These indicate that all the highly similar skeleton model groups lie on manifold but are well separated from each other. From the same figure, it also outperforms MDS retrieval as it is not a manifold learning technique.

7.3 Discussion and Evaluation

7.3.1 Strengths

7.3.1.1 Accuracy

In the first part of the research, we have developed a feature extraction (TPR) and matching method that can handle 3D articulated geometry models. The method uses topological points and topological rings as features. These features are automatically extracted from 3D mesh based on content analysis. We have shown empirically that the method (TPR) outperforms Multi-resolution reeb graph (MRG) [17] in term of accuracy on a small database (Chapter 4) and a large database (Section 7.2). The improvement is resulted from the fact that we have captured a lot of

geometric (local and global) features, while MRG only captures local features.

In the second part of the research, we have developed an embedding retrieval framework based on manifold learning technique, Diffusion Map. It learns the manifolds that these models lie on. By projecting them on the first few eigenvectors, it is able to maximize the inter-class distances between these model groups, achieving great results. In Figure 7.5, we put together all the precision and recall curves. We see that our method TPR performs better than MRG before and after using embedding retrieval. It should be noted that all these results have been presented and discussed in Chapter 5.

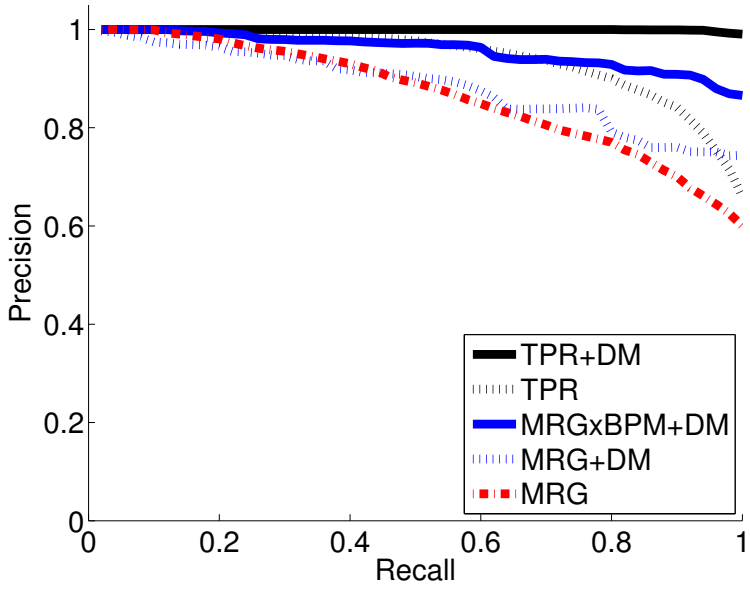


Figure 7.5: Precision and Recall of applying Embedding Retrieval to TPR and MRG method on 1020 models.

7.3.1.2 Efficiency

We have shown empirically that our method (TPR) outperforms Multi-resolution reeb graph (MRG) [17] in term of speed as well. The similarity measure is a metric distance and so supports indexing. From our experiments, if we simply want to find the first few relevant results of a query, the nearest neighbor search is able to fast retrieve by pruning unnecessary branches of the indexing tree.

As shown in second part of the research and in our experiments, the retrieval

framework only requires to invoke those graph and bag-based matching algorithms on 15% of the database. This largely reduces the retrieval times. From the time curve, it also shows that embedding retrieval can avoid the curse of dimensionality, by using a manifold learning technique to reduce the intrinsic dimension.

7.3.2 Weaknesses

After confirming the strengths of our work, we also observe many short-comings of our methods. We would like to discuss them here. It also allows us to define future improvements and research directions.

7.3.2.1 Feature representation, extraction and matching

Though the proposed feature extraction algorithm is reliable to our database, it may not work fully on every kind of models. First of all, we have proposed a *Critical Point Analysis* (Section 4.3.4) and *Topological Point Selection* (Section 4.5.1) to alleviate the extraneous critical points and slicing direction problems, respectively. The idea is based on simple filtering and majority voting. However, when the model has a very noise surface, these methods may still mistakenly extract the noisy critical points. Second, both of these two techniques are based on Level Set Diagram (LSD), a skeleton extraction algorithm. This also implies an assumption that the method works on models with tubular parts. If the models consist of many flat, round or ball (non-tubular) shapes, we expect that the algorithm may not extract meaningful features. One of the solutions would be using mesh segmentation. However, mesh segmentation usually involves precise segmentations (patches, parts) and boundary smoothing. This also suggests that these algorithms may be too slow for online feature extraction.

Though our feature extraction (TPR) and matching algorithm ourperform MRG on accuracy and speed, there are a lots of drawbacks as well. The main problem is its inflexibility. Currently, our method does not support model with arbitrary genus, non-manifold or non-closed models, but MRG does. The reason is also related to the previous problem that TPR is built on LSD. During our design of TPR, we exclude meshes with genus greater than 0 to simplify the feature extraction problem,

therefore, it is currently undefined on higher genus meshes.

Our feature representation is also restricted as well. In order to give better descriptive power, we have extracted both local and global features. Global features as shown are good for discriminating highly similar skeleton models, but at the same time, they cannot be extended to subpart matching. In fact, many existing graph-based ([45]) and bag-based ([16], [13], [14]) matching methods can be easily extended to sub-part matching because their features are relatively local. This suggests that it is challenging to improve our feature representation to support subpart matching. As discussed in an earlier section (Section 7.2), though geodesic distance is generally believed to be stable towards articulation, our global features also suffer from slight misalignment problem. Further improvement has to be researched.

In our experiments, though the feature matching method Earth Mover Distance (EMD) works fine, the complexity is still high $O(n^3 \log n)$. The efficiency of our work is resulted from the compact representation. However, when a model contains many branches (subparts), it would become a very slow task. In our experiment, EMD requires on average $1ms$ to compare two models. This is fast compared to the existing *Bag of features* approach; however, it is still relatively slow compared to the *Single feature vector* approach (avg $\leq 1\mu s$). Consider a database containing millions of records, $1ms$ would render the whole system irresponsive, even after the application of fast pruning search.

7.3.2.2 Embedding Retrieval

The proposed embedding retrieval framework is based on the observation and analysis that graph and bag-based matching will lead to Euclidean-like matching. Slight misalignment will cause large class variance. To handle that we use a manifold learning technique to conduct dimension reduction and use Nyström extension for fast retrieval. Here, we would like to discuss some drawbacks and disadvantages of the framework.

First, our framework assumes that the data in question lie on manifold(s). In our experiments, because both TPR and MRG are built by partitioning a metric function (geodesic and integral geodesic, respectively), the similarity measure produces large

variance in the same group, which matches the criteria of manifold data. If the data are non-manifold, however, there is no guarantee that our method would work. It may even degrade retrieval performance because it preserves close distances only and ignore far distances entirely.

Another assumption is that there must be lots of data such that a neighborhood can be defined. This is required for all manifold learning techniques. In Diffusion Map, it is required to carry out diffusion. In fact, the reason that these methods can learn various manifolds and maximize manifold distances is based on the fact that there are sufficient objects in the database to allow such diffusion and to define the neighborhood of the manifold. However, this may not always be feasible especially on small databases.

When the data is not manifold-like or there is insufficient data, Multi-Dimensional Scaling (MDS) would be the best choice because it preserves all (close and far) distances in the embedding space. When the number of dimensions increases, MDS can better preserve all pairwise distances, leading to a retrieval performance closer to that of the original similarity measure (metric or non-metric). However the required dimensionality may be over hundreds, and the method will surely suffer from the curse of dimensionality. In either case, MDS and our retrieval framework are both approximate methods and will introduce false dismissal.

In order to preserve retrieval accuracy, we have proposed an alignment step to project a query coordinate from Nyström embedding to eigensolver embedding. To pre-compute eigensolver embedding, we use sparse eigensolver. However, eigensolver is well-known to be memory intensive and slow, especially when the database is huge. Currently, our method assumes a static database and dynamic query. Whenever a new object is added to the database, it has to solve the eigensolver embedding again. An approximation would be made to project the new object as query and use our alignment step to obtain an approximate coordinate in the eigensolver embedding, but this is only an approximation. When the number of new objects increases, it still requires to recompute the eigensolver embedding to ensure the retrieval accuracy.

7.4 Application on 3D Motion

In the third part of the research, we have applied the idea of bag-of-features to represent 3D motion and developed corresponding matching algorithm and similarity measure. The main contribution is to subdivide a motion into pieces and capture features for each piece. As shown in our experiments, it also outperforms existing methods, Uniform Scaling [9] and Dynamic Time Warping [8]. The method also defines a metric similarity measure based on Earth Mover Distance. It suggests that a distance-based indexing technique is applicable. Though we have not applied indexing structure for retrieval in the current work, the method is also comparatively faster than Uniform Scaling and Dynamic Time Warping.

Though the 3D motion retrieval method is fast and accurate, there are also drawbacks of the method. First, our method allows local and global shifting at the same time. However, it also allows strayed matching. This is not useful for "copying and pasting" different motions together because concatenating two motions requires an exact match at the starting or ending segment of the motions. Currently, because we consider matching of individual joints only, frame synchronization of different joints is not considered. Further analysis is needed to verify the importance of frame synchronization to motion similarity search.

With regard to the personal conversations with Prof. Eamonn Koegh, the author of both Dynamic Time Warping [8] and Uniform Scaling [9], our current feature representation may not truly represent the original motion content because it only makes use of the start/end point of a time series segment. Due to this reason, the similarity measure is a metric with regard to the extracted features only. To solve the issue, a better solution is to use linear regression (find the best fitting line). It would give a better representation and retrieval results [124].

Apart from accuracy issue, the feature representation may also suffer from the curse of dimensionality. In order to give a good indexing scheme that fully describes the raw data (original time series) and reduce the effect of the curse of dimensionality, it would be better to use a filter-and-refine technique (Section 2.4.2.1) for fast pruning.

Another important issue is that we have manually defined a lot of joint weights

(29 weights) in order to combine all EMD distances into one. Though the definition of weights is intuitive and works fine on our small (115 motions) datasets, the portability of these weights to other datasets are still untested. Another concern is that these weights may have semantic effect as well. For example, a man playing judo would have large movement on legs, e.g. kicks. But for a baby, the little activities of fingers and toes may be more important. These suggest that more research would be done to relate these parameters to various activities, and adapting machine learning approach to learn or optimize these parameters would be one of the possible works.

7.5 Summary

In this research study, we focus ourselves on the reliability and efficiency of developing retrieval techniques for 3D articulated geometry models. We have proposed an efficient matching method. It uses compact features and Earth Mover Distance for matching. The matching method automatically analyze models, extract topological and geometric features and is fully automatic without human annotation. We have also defined metric measure that can support indexing. To avoid the curse of dimensionality, we have also developed an embedding retrieval framework. It not only improve efficiency but also accuracy in one goal. With respect to Section 3.1, we believe that we have fulfilled all of our research goals.

However, as discussed in the chapter, we have also observed many shortcomings of our methods. These, however, provide challenges and new research directions as we discuss in the next chapter.

Chapter 8

Future Work and Conclusion

8.1 Future Work

8.1.1 Extension to Current Works

In the previous chapter, we have contrasted the strengths and weaknesses of our methods. In this section, we would like to discuss various extensions that may overcome those shortcomings.

1. ***Feature Extraction***

   Currently we apply critical point analysis to capture topological points as features. The method is based on majority voting to select reliable critical points. The method is fast, but it may still suffer from noise. A better critical point extraction could be developed. Recently, there are a lot of mesh segmentation methods being proposed. One closely related option is to employ the "Core Extraction" technique [125]. The method uses Multi-Dimensional Scaling (MDS) to pose-normalize a model and extract outlying extreme points. Since these outlying extreme points match the definition of our topological points and are extracted in the transformed domain where noise is filtered, they should be more reliable. Similarly, to capture topological rings, we would employ a mesh segmentation technique and define them as the boundaries of different segments. However, since these mesh segmentation algorithms require precise boundary decision and smoothing, they are usually slow. Balancing the effi-

ciency and accuracy become one of the interesting but challenging directions.

We also extract three surface distributions (curvatures, areas and thickness) as geometric features (Section 4.5.5). These distribution bands are computed from geodesic. Though geodesic is assumed to be articulation invariant, it is shown in Section 5.2 that it still suffers from slight misalignment leading to large variance in the data. To improve that, one of the ideas is to use better surface metric (e.g., part-aware metric [50]) instead of geodesic or integral geodesic. However, as long as the similarity measure uses Euclidean distance, it may also suffer from the slight misalignment problem and the curse of dimensionality.

2. ***Embedding Retrieval***

   In our embedding retrieval framework, we have optimized parameters for retrieval purposes. In particular, we limit the number of dimensions to 10 to avoid the curse of dimensionality. Since our framework converts similarity measure into Euclidean coordinates, it would be better to use Local Sensitive Hashing (LSH) [69] instead of Kd-tree. LSH has been shown to scale well with high dimensionality theoretically and practically, which works in vector space. By applying LSH, we would expect to be able to raise the 10-dimension limit. This would possibly lead to a higher retrieval performance because more information can be retained in the embedding space. It should be noted that, currently, manifold learning is still under active research. We do not recognize any theory in the area that connects dimensionality to retrieval performance. It would be an interesting direction for research too.

   In the experiments, we have created a large database of 1020 models. This is relatively large compared to existing works because they are all high quality models focusing on both similar and dissimilar skeletons. However, compared to other multimedia retrieval systems (of size $\geq$ 100000), it is only a small database. Currently we have applied Nyström extension for fast retrieval. To preserve retrieval accuracy we have to pre-compute embedding using eigensolver. Since eigensolver is always limited by memory, this restricts the ap-

plication of the framework. One possibility is to employ out-of-core eigensolver [126] to obtain reliable embedding. Another idea is to use Nyström extension also for such pre-computation. To achieve the best accuracy, we can use Nyström extension with as many landmarks as possible, up to the limit of memory constraint. Though it is still an approximation, the approximation should be a good one because Nyström extension has been shown to converge given sufficient samples.

In our current project, we strive to provide an unsupervised content-based retrieval method for 3D articulated geometry models. However, it would be possible to apply supervised method or even machine learning approach to achieve better performance. Example includes Self-Organisation Map (SOM) and Support Vector Machine (SVM).

3. ***Motion Retrieval***

   We have applied our bag-based matching knowledge into motion matching. However, as pointed out by Prof. Keogh, the method may not be a metric with respect to the raw motion data, which is essential for precise matching. One of the possible ideas is to use "linear regression" (line fitting) instead of "linear interpolation" (start-point, as in our case) as it conveys more information for lower-bound design [124]. In fact, many successful lower-bounding techniques [8], [9] have been designed based on similar idea. Recently, Wichterich et al. [127] also propose an lower bound technique for Earth Mover Distance. It would be a very good idea to follow up this direction and develop more reliable motion matching and fast search technique.

8.1.2 Possible Future Research Directions

In this section, we would like to discuss some closely related research areas that we would like to explore in the future.

1. ***Subpart Matching***

   Subpart matching is a new research area that is less frequently discussed or explored. The area is important because, it allows the user to provide a part

(e.g., hand) and to search for the whole model (e.g., boy). It has a higher usability and practicality than a whole model retrieval system. However, it is also a challenging task. On the one hand, developing such matching algorithms usually require mesh segmentation techniques, which are still under active research. On the other hand, the matching algorithms are usually slow and non-metric. The reason that these algorithms are non-metric because subpart matching closely follows human perception. Human perception is based on subpart coherence, which is also non-metric in nature. In fact, Earth Mover Distance can also be used for subpart matching. However, by doing so, the similarity measure becomes non-metric as well because it violates the constraint of equal weights. All these indicate that, similar to whole model matching, no metric indexing technique is applicable. Efficiency and scalability are some of the challenges of the area.

In this research study, we observe and suggest that graph and bag-based matching methods project data on manifolds. This is useful to convert similarity measure from non-metric space into metric vector space. However, it is still an open question if it can be applied on subpart matching. To do so, one important question has to be asked: how non-metric space and (non-linear) manifold space can be related (or are there any relationship at all)?. It should be noted that manifold learning is still in active research. Researching in these areas would be challenging but also fruitful.

2. ***Mesh Correspondence***

   As mentioned in the literature, pose-normalization technique is also a new and challenging area. Currently most of the work transforms the mesh into embedding space so that the model can be normalized. Though transformation to low-dimensional space may lose important geometric features, it also opens up a new area on "correspondence" analysis [105]. If correspondence can be established automatically, surface detail transfer would also be possible and thus also useful for model comparison. In fact, latest research works are also moving in this direction [128].

3. ***Semi-Supervised Learning and Multi-Modal Analysis***

   In our current work, we have discussed the use of manifold learning to maximize inter-class distance. Our use of manifold learning can be considered unsupervised learning. It is interesting that manifold learning is also applicable for semi-supervised learning. The term "semi-supervised" means that users can provide a few initial inputs and the system learns (diffuses) the rest. The whole scheme is called transduction. It has been useful for segmentation [129] and relevancy feedback [99]. We have used manifold learning in our retrieval system in the spirit of segmentation. It would be interesting to see if it is beneficial to relevancy feedback of 3D model research in the future.

   We have also developed an augmented kernel method to pull different manifolds of the same group together. Currently, we rely on another similarity measure to establish such shortcut edges. However, it would also be possible to use other kind of analysis to provide such information. This is called multi-modal retrieval and has been frequently used in video analysis. The idea is to use sounds, images, videos, as well as text to provide a more meaningful retrieval scheme [130]. The same idea would also be applied to 3D articulated geometry models matching. For example, some information from an ontology would bridge the semantic gap between machine and human perception.

4. ***Computer Vision and 2D/3D Motion Classification***

   There are many on-going motion researches in computer vision that investigate motion tracking, estimation, matching and classification in video sequences. Some recent works include [131] (classification of motions based on robust appearance and motion descriptor), [132] (human pose estimation using skeletal 2D models), [133] (video-based gait kinematics) and [134] (human motion classification using image-based rendering and reconstruction) etc. Though our project focuses on the 3D motion matching using 3D mocap data, we find that many of these techniques and ideas share across 2D and 3D motion analysis. For example, [133] use manifold learning methods to model whole, upper and lower body part motions. The models are trained by CMU mocap data, that we have used in our experiments, to estimate gait kinematics from video.

Classification works also require similarity measures that are based on bag-of-feature [131] and histogram descriptor [134], whilst classification is achieved by Support Vector Machine. We believe that more future works can be discovered by exploring different robust computer vision techniques. At the same time, we also believe that our research may be beneficial to computer vision research when an existing matching methods on mocap data is required.

8.2 Conclusion

In this research, we have developed algorithms for 3D articulated geometry models, covering feature extraction, feature matching, indexing and fast search methods. All these methods outperform existing works in reliability and efficiency and have been demonstrated through various experiments. Our idea of converting restricted matching to bag-based matching has also been applied to 3D motion. A feature extraction and matching algorithm has also been proposed that outperforms existing works. We have also connected graph and bag-based matching methods of 3D models to the area of manifold learning. We believe that we have fulfilled all our research goals, generated new knowledges in the area and also discovered many potential research directions.

References

[1] T. Funkhouser, M. Kazhdan, P. Shilane, P. Min, W. Kiefer, A. Tal, S. Rusinkiewicz, and D. Dobkin, "Modeling by example," *ACM Trans. Graph.*, vol. 23, no. 3, pp. 652–663, 2004.

[2] L. Kovar, M. Gleicher, and F. Pighin, "Motion graphs," in *ACM SIGGRAPH 2002*, vol. 21, (New York, NY, USA), pp. 473–482, July 2002.

[3] J. Lee, J. Chai, P. S. A. Reitsma, J. K. Hodgins, and N. S. Pollard, "Interactive control of avatars animated with human motion data," in *SIGGRAPH'02: Proceedings of the 29th annual conference on Computer graphics and interactive techniques*, (New York, NY, USA), pp. 491–500, ACM Press, 2002.

[4] H. Sundar, D. Silver, N. Gagvani, and S. J. Dickinson, "Skeleton based shape matching and retrieval," in *SMI '03: Proceedings of the Shape Modeling International 2003*, (Washington, DC, USA), p. 130, IEEE Computer Society, 2003.

[5] A. Tal and E. Zuckerberger, "Mesh retrieval by components," in *International Conference on Computer Graphics Theory and Applications*, pp. 142–149, 2006.

[6] M. F. Demirci, R. H. van Leuken, and R. C. Veltkamp, "Indexing through laplacian spectra," *Comput. Vis. Image Underst.*, vol. 110, no. 3, pp. 312–325, 2008.

[7] R. H. van Leuken, O. Symonova, R. C. Veltkamp, and R. de Amicis, "Complex fiedler vectors for shape retrieval," in *Structural, Syntactic, and Statistical*

Pattern Recognition, Joint IAPR International Workshop, SSPR and SPR 2008, pp. 167–176, 2008.

[8] E. Keogh, "Exact indexing of dynamic time warping," in *VLDB*, pp. 406–417, 2002.

[9] E. Keogh, T. Palpanas, V. B. Zordan, D. Gunopulos, and M. Cardle, "Indexing large human-motion databases.," in *VLDB*, pp. 780–791, 2004.

[10] T. Tung and F. Schmitt, "Augmented reeb graphs for content-based retrieval of 3D mesh models," in *SMI '04: Proceedings of the Shape Modeling International 2004*, (Washington, DC, USA), pp. 157–166, IEEE Computer Society, 2004.

[11] D. Bespalov, A. Shokoufandeh, W. C. Regli, and W. Sun, "Scale-space representation of 3D models and topological matching," in *SM '03: Proceedings of the eighth ACM symposium on Solid modeling and applications*, (New York, NY, USA), pp. 208–215, ACM, 2003.

[12] D. Bespalov, W. C. Regli, and A. Shokoufandeh, "Local feature extraction and matching partial objects," *Computer-Aided Design*, vol. 38, pp. 1020–1037, Sept. 2006.

[13] J. Tierny, J.-P. Vandeborre, and M. Daoudi, "Reeb chart unfolding based 3D shape signatures," in *Eurographics*, (Prague, Czech Republic), pp. 13–16, 2007.

[14] M. R. Ruggeri and D. Saupe, "Isometry-invariant matching of point set surfaces," in *Proceedings Eurographics 2008 Workshop on 3D Object Retrieval*, 2008.

[15] K.-L. Tam, R. W. H. Lau, and C.-w. Ngo, "Deformable geometry model matching by topological and geometric signatures," in *ICPR (3)*, pp. 910–913, 2004.

[16] N. D. Cornea, M. F. Demirci, D. Silver, A. Shokoufandeh, S. J. Dickinson, and P. B. Kantor, "3D object retrieval using many-to-many matching of curve

skeletons," in *SMI '05: Proceedings of the International Conference on Shape Modeling and Applications 2005*, (Washington, DC, USA), pp. 368–373, IEEE Computer Society, 2005.

[17] M. Hilaga, Y. Shinagawa, T. Kohmura, and T. L. Kunii, "Topology matching for fully automatic similarity estimation of 3D shapes," in *SIGGRAPH '01: Proceedings of the 28th annual conference on Computer graphics and interactive techniques*, (New York, NY, USA), pp. 203–212, ACM Press, 2001.

[18] K. Zhang and J. T. Kwok, "Density-weighted nyström method for computing large kernel eigensystems," *Neural Comput.*, vol. 21, no. 1, pp. 121–146, 2009.

[19] G. K. L. Tam, "Matching and retrieval of 3D deformable models," Master's thesis, City University of Hong Kong, Nov. 2004.

[20] G. K. L. Tam and R. W. H. Lau, "Deformable model retrieval based on topological and geometric signatures," *IEEE Transactions on Visualization and Computer Graphics*, vol. 13, no. 3, pp. 470–482, 2007.

[21] G. K. Tam and R. W. Lau, "Embedding retrieval of 3D articulated geometry models." currently under review, submitted to ACM Multimedia 2009, 2009.

[22] G. K. L. Tam, Q. Zheng, M. Corbyn, and R. W. H. Lau, "Motion retrieval based on energy morphing," in *ISM '07: Proceedings of the Ninth IEEE International Symposium on Multimedia*, (Washington, DC, USA), pp. 210–220, IEEE Computer Society, 2007.

[23] M. Kazhdan, T. Funkhouser, and S. Rusinkiewicz, "Shape matching and anisotropy," in *SIGGRAPH '04: ACM SIGGRAPH 2004 Papers*, (New York, NY, USA), pp. 623–629, ACM, 2004.

[24] M. Elad, A. Tal, and S. Ar, "Content based retrieval of VRML objects: an iterative and interactive approach," in *Proceedings of the sixth Eurographics workshop on Multimedia 2001*, (New York, NY, USA), pp. 107–118, Springer-Verlag New York, Inc., 2002.

[25] T. Funkhouser and P. Shilane, "Partial matching of 3D shapes with priority-driven search," in *SGP'06: Proceedings of the fourth Eurographics symposium on Geometry processing*, (Aire-la-Ville, Switzerland, Switzerland), pp. 131–142, Eurographics Association, 2006.

[26] R. Ohbuchi and T. Takei, "Shape-similarity comparison of 3D models using alpha shapes," in *PG '03: Proceedings of the 11th Pacific Conference on Computer Graphics and Applications*, (Washington, DC, USA), pp. 293–302, IEEE Computer Society, 2003.

[27] R. Osada, T. Funkhouser, B. Chazelle, and D. Dobkin, "Matching 3D models with shape distributions," in *SMI '01: Proceedings of the International Conference on Shape Modeling and Applications*, (Washington, DC, USA), pp. 154–166, IEEE Computer Society, 2001.

[28] M. Kazhdan, B. Chazelle, D. Dobkin, T. Funkhouser, and S. Rusinkiewicz, "A reflective symmetry descriptor for 3D models," *Algorithmica*, vol. 38, no. 1, pp. 201–225, 2003.

[29] T. D. Joachim, J. Giesen, and S. Goswami, "Shape segmentation and matching with flow discretization," in *In Proc. Workshop on Algorithms and Data Structures*, pp. 25–36, 2003.

[30] J. W. H. Tangelder and R. C. Veltkamp, "Polyhedral model retrieval using weighted point sets," in *SMI '03: Proceedings of the Shape Modeling International 2003*, (Washington, DC, USA), p. 119, IEEE Computer Society, 2003.

[31] M. Yu, I. Atmosukarto, W. K. Leow, Z. Huang, and R. Xu, "3D model retrieval with morphing-based geometric and topological feature maps," in *Proc. IEEE Conf. on Computer Vision and Pattern Recognition*, pp. 656–661, 2003.

[32] D. V. Vranić and D. Saupe, "3D shape descriptor based on 3D fourier transform," in *Proc. of ECMCS*, pp. 271–274, Sept. 2001.

[33] E. Paquet and M. Rioux, "Content-based access of VRML libraries," in *MINAR '98: Proceedings of the IAPR International Workshop on Multimedia In-*

formation Analysis and Retrieval, (London, UK), pp. 20–32, Springer-Verlag, 1998.

[34] M. Novotni and R. Klein, "3D zernike descriptors for content based shape retrieval," in *SM '03: Proceedings of the eighth ACM symposium on Solid modeling and applications*, (New York, NY, USA), pp. 216–225, ACM, 2003.

[35] T. Funkhouser, P. Min, M. Kazhdan, J. Chen, A. Halderman, D. Dobkin, and D. Jacobs, "A search engine for 3D models," *ACM Trans. Graph.*, vol. 22, no. 1, pp. 83–105, 2003.

[36] M. Kazhdan, T. Funkhouser, and S. Rusinkiewicz, "Rotation invariant spherical harmonic representation of 3D shape descriptors," in *SGP '03: Proceedings of the 2003 Eurographics/ACM SIGGRAPH symposium on Geometry processing*, (Aire-la-Ville, Switzerland, Switzerland), pp. 156–164, Eurographics Association, 2003.

[37] D.-Y. Chen, X.-P. Tian, Y.-T. Shen, and M. Ouhyoung, "On visual similarity based 3D model retrieval," *Computer Graphics Forum*, vol. 22, no. 3, pp. 223–232, 2003. Eurographics 2003 Conference Proceedings.

[38] R. Ohbuchi, M. Nakazawa, and T. Takei, "Retrieving 3D shapes based on their appearance," in *MIR '03: Proceedings of the 5th ACM SIGMM international workshop on Multimedia information retrieval*, (New York, NY, USA), pp. 39–45, ACM, 2003.

[39] P. Min, J. A. Halderman, M. Kazhdan, and T. A. Funkhouser, "Early experiences with a 3D model search engine," in *Web3D '03: Proceedings of the eighth international conference on 3D Web technology*, (New York, NY, USA), pp. 7–18, ACM, 2003.

[40] F. Lazarus and A. Verroust, "Level set diagrams of polyhedral objects," in *SMA '99: Proceedings of the fifth ACM symposium on Solid modeling and applications*, (New York, NY, USA), pp. 130–140, ACM, 1999.

[41] S. Biasotti, D. Giorgi, M. Spagnuolo, and B. Falcidieno, "Size functions for comparing 3D models," *Pattern Recognition*, vol. 41, no. 9, pp. 2855–2873, 2008.

[42] H. Blum, "A transformation for extracting new descriptors of shape," *Models for the Perception of Speech and Visual Form*, pp. 362–380, 1967.

[43] R. Kimmel and J. Sethian, "Computing geodesic paths on manifolds," *National Academy of Sciences of the USA*, vol. 95, pp. 8431–8435, 1998.

[44] M. Novotni and R. Klein, "Computing geodesic distances on triangular meshes," in *The 10-th International Conference in Central Europe on Computer Graphics, Visualization and Computer Vision'2002 (WSCG'2002)*, Feb. 2002.

[45] S. Biasotti, S. Marini, M. Spagnuolo, and B. Falcidieno, "Sub-part correspondence by structural descriptors of 3D shapes," *Computer-Aided Design*, vol. 38, pp. 1002–1019, Sept. 2006.

[46] G. K. L. Tam, R. W. H. Lau, and C.-w. Ngo, "Deformable geometry model matching using bipartite graph," in *Proc. Computer Graphics International*, pp. 335–342, 2004.

[47] I. Assent, A. Wenning, and T. Seidl, "Approximation techniques for indexing the earth mover's distance in multimedia databases," in *ICDE '06: Proceedings of the 22nd International Conference on Data Engineering*, (Washington, DC, USA), pp. 11–11, IEEE Computer Society, 2006.

[48] R. Gal, A. Shamir, and D. Cohen-Or, "Pose-oblivious shape signature," *IEEE Transactions on Visualization and Computer Graphics*, vol. 13, no. 2, pp. 261–271, 2007.

[49] A. Ion, N. M. Artner, G. Peyre, S. B. Lopez Marmol, W. G. Kropatsch, and L. D. Cohen, "3D shape matching by geodesic eccentricity," in *Computer Vision and Pattern Recognition Workshops*, pp. 1–8, 2008.

[50] R. Liu, H. Zhang, A. Shamir, and D. Cohen-Or, "A part-aware surface metric for shape analysis," *Computer Graphics Forum (Special Issue of Eurographics 2009)*, vol. 28, no. 2, pp. 397–406, 2009.

[51] M. Reuter, F.-E. Wolter, and N. Peinecke, "Laplace-beltrami spectra as "shape-dna" of surfaces and solids," *Computer-Aided Design*, vol. 38, no. 4, pp. 342–366, 2006.

[52] A. Elad and R. Kimmel, "On bending invariant signatures for surfaces," *IEEE Transactions on Pattern Analysis and Machine Intelligence*, vol. 25, no. 10, pp. 1285–1295, 2003.

[53] T. F. Cox and M. A. Cox, *Multidimensional Scaling*. Chapman and Hall, London, 1994.

[54] V. Jain and H. Zhang, "Shape-based retrieval of articulated 3D models using spectral embedding.," in *Geometric Modeling and Processing*, pp. 299–312, 2006.

[55] R. M. Rustamov, "Laplace-beltrami eigenfunctions for deformation invariant shape representation," in *SGP '07: Proceedings of the fifth Eurographics symposium on Geometry processing*, (Aire-la-Ville, Switzerland, Switzerland), pp. 225–233, Eurographics Association, 2007.

[56] K.-S. Goh, B. Li, and E. Chang, "Dyndex: a dynamic and non-metric space indexer," in *MULTIMEDIA 02: Proceedings of the tenth ACM international conference on Multimedia*, (New York, NY, USA), pp. 466–475, ACM, 2002.

[57] D. W. Jacobs, D. Weinshall, and Y. Gdalyahu, "Classification with nonmetric distances: Image retrieval and class representation," *IEEE Transactions on Pattern Analysis and Machine Intelligence*, vol. 22, no. 6, pp. 583–600, 2000.

[58] A. Tversky, "Features of similarity," *Psychological Review*, vol. 84, pp. 327–352, July 1977.

[59] S. Biasotti, S. Marini, M. Mortara, G. Patané, M. Spagnuolo, and B. Falcidieno, "3D shape matching through topological structures," in *Lecture Notes in Computer Science*, vol. 2886, pp. 194–203, 2003.

[60] C. Böhm, S. Berchtold, and D. A. Keim, "Searching in high-dimensional spaces: Index structures for improving the performance of multimedia databases," *ACM Comput. Surv.*, vol. 33, no. 3, pp. 322–373, 2001.

[61] G. R. Hjaltason and H. Samet, "Properties of embedding methods for similarity searching in metric spaces," *IEEE Transactions on Pattern Analysis and Machine Intelligence*, vol. 25, no. 5, pp. 530–549, 2003.

[62] G. R. Hjaltason and H. Samet, "Index-driven similarity search in metric spaces (survey article)," *ACM Trans. Database Syst.*, vol. 28, no. 4, pp. 517–580, 2003.

[63] D. P. Huttenlocher, G. A. Klanderman, and W. A. Rucklidge, "Comparing images using the hausdorff distance," *IEEE Transactions on Pattern Analysis and Machine Intelligence*, vol. 15, no. 9, pp. 850–863, 1993.

[64] A. W.-c. Fu, P. M.-s. Chan, Y.-L. Cheung, and Y. S. Moon, "Dynamic vp-tree indexing for n-nearest neighbor search given pair-wise distances," *The VLDB Journal*, vol. 9, no. 2, pp. 154–173, 2000.

[65] T. Bozkaya and M. Ozsoyoglu, "Indexing large metric spaces for similarity search queries," *ACM Trans. Database Syst.*, vol. 24, no. 3, pp. 361–404, 1999.

[66] P. Ciaccia, M. Patella, and P. Zezula, "M-tree: An efficient access method for similarity search in metric spaces," in *VLDB'97, Proceedings of 23rd International Conference on Very Large Data Bases, August 25-29, 1997, Athens, Greece*, pp. 426–435, Morgan Kaufmann, 1997.

[67] P. Zezula, P. Savino, G. Amato, and F. Rabitti, "Approximate similarity retrieval with m-trees," *The VLDB Journal*, vol. 7, no. 4, pp. 275–293, 1998.

[68] P. Indyk, "Nearest neighbors in high-dimensional spaces," in *Handbook of Discrete and Computational Geometry (2nd edition) J. E. Goodman and J. O'Rourke, editors.*, CRC Press LLC., to appear.

[69] A. Gionis, P. Indyk, and R. Motwani, "Similarity search in high dimensions via hashing," in *VLDB ́99: Proceedings of the 25th International Conference on Very Large Data Bases*, (San Francisco, CA, USA), pp. 518–529, Morgan Kaufmann Publishers Inc., 1999.

[70] J. Bourgain, "On lipschitz embedding of finite metric spaces in hilbert space," *Israel Journal of Mathematics*, vol. 52, pp. 46–52, Mar. 1985.

[71] H. Gabriela and F. Martin, "Cluster-preserving embedding of proteins," tech. rep., Department of Computer Science, Rutgers University, 1999.

[72] C. Faloutsos and K.-I. Lin, "Fastmap: a fast algorithm for indexing, data-mining and visualization of traditional and multimedia datasets," in *SIGMOD '95: Proceedings of the 1995 ACM SIGMOD international conference on Management of data*, (New York, NY, USA), pp. 163–174, ACM, 1995.

[73] J. T. L. Wang, X. Wang, D. Shasha, and K. Zhang, "Metricmap: an embedding technique for processing distance-based queries in metric spaces," *Systems, Man, and Cybernetics, Part B, IEEE Transactions on*, vol. 35, no. 5, pp. 973–987, 2005.

[74] V. de Silva and J. B. Tenenbaum, "Sparse multidimensional scaling using landmark points," tech. rep., Stanford University, 2004.

[75] J. C. Platt, "Fastmap, metricmap, and landmark mds are all nyström algorithms," in *10th International Workshop on Artificial Intelligence and Statistics*, pp. 261–268, 2005.

[76] D. Donoho, "High-dimensional data analysis: The curses and blessings of dimensionality." Lecture delivered at the conference "Math Challenges of the 21st Century" held by the American Math. Society organised in Los Angeles, August 6-11, 2000.

[77] F. Korn, B.-U. Pagel, and C. Faloutsos, "On the 'dimensionality curse' and the 'self-similarity blessing'," *IEEE Trans. on Knowl. and Data Eng.*, vol. 13, no. 1, pp. 96–111, 2001.

[78] S. T. Roweis and L. K. Saul, "Nonlinear dimensionality reduction by locally linear embedding," *Science*, vol. 290, no. 5500, pp. 2323–2326, 2000.

[79] J. B. Tenenbaum, V. d. Silva, and J. C. Langford, "A global geometric framework for nonlinear dimensionality reduction," *Science*, vol. 290, no. 5500, pp. 2319–2323, 2000.

[80] L. Chen and X. Lian, "Efficient similarity search in nonmetric spaces with local constant embedding," *IEEE Trans. on Knowl. and Data Eng.*, vol. 20, no. 3, pp. 321–336, 2008.

[81] V. Roth, J. Laub, J. Buhmann, and K. Muller, "Going metric: Denoising pairwise data," *NIPS*, pp. 817–824, 2002.

[82] M. Vlachos, M. Hadjieleftheriou, D. Gunopulos, and E. Keogh, "Indexing multi-dimensional time-series with support for multiple distance measures," in *KDD '03: Proceedings of the ninth ACM SIGKDD international conference on Knowledge discovery and data mining*, (New York, NY, USA), pp. 216–225, ACM Press, 2003.

[83] T. Skopal, "On fast non-metric similarity search by metric access methods," in *EDBT*, pp. 718–736, 2006.

[84] V. Athitsos, M. Hadjieleftheriou, G. Kollios, and S. Sclaroff, "Query-sensitive embeddings," in *SIGMOD '05: Proceedings of the 2005 ACM SIGMOD international conference on Management of data*, (New York, NY, USA), pp. 706–717, ACM, 2005.

[85] V. Athitsos, J. Alon, S. Sclaroff, and G. Kollios, "Boostmap: An embedding method for efficient nearest neighbor retrieval," *IEEE Trans. Pattern Anal. Mach. Intell.*, vol. 30, no. 1, pp. 89–104, 2008.

[86] Y. Rubner, C. Tomasi, and L. J. Guibas, "The earth mover's distance as a metric for image retrieval," *Int. J. Comput. Vision*, vol. 40, no. 2, pp. 99–121, 2000.

[87] X. Ni, M. Garland, and J. C. Hart, "Fair morse functions for extracting the topological structure of a surface mesh," in *SIGGRAPH'04: ACM SIGGRAPH 2004 Papers*, (New York, NY, USA), pp. 613–622, ACM Press, 2004.

[88] M. Mortara and G. Patané, "Affine-invariant skeleton of 3D shapes," in *SMI '02: Proceedings of the Shape Modeling International 2002 (SMI'02)*, (Washington, DC, USA), pp. 245–252, IEEE Computer Society, 2002.

[89] M. Meyer, M. Desbrun, P. Schröder, and A. H. Barr, "Discrete differential-geometry operators for triangulated 2-manifolds," in *VisMath*, 2002.

[90] S. Popinet, "Gnu triangulated surface library." Website, 2006. `http://gts.sourceforge.net/index.html`.

[91] J. E. Goodman and J. O'Rourke, eds., *Handbook of discrete and computational geometry*. Boca Raton, FL, USA: CRC Press, Inc., 2004.

[92] R. R. Coifman and S. Lafon, "Diffusion maps," *Applied and Computational Harmonic Analysis*, vol. 21, pp. 5–30, July 2006.

[93] V. de Silva and J. B. Tenenbaum, "Global versus local methods in nonlinear dimensionality reduction," in *NIPS*, pp. 721–728, 2003.

[94] S. S. Lafon, *Diffusion Maps and Geometric Harmonic*. PhD Thesis, Yale University, May 2004.

[95] J. Shi and J. Malik, "Normalized cuts and image segmentation," *Pattern Analysis and Machine Intelligence, IEEE Transactions on*, vol. 22, pp. 888–905, Aug 2000.

[96] R. Liu and H. Zhang, "Segmentation of 3D meshes through spectral clustering," in *PG 04: Proceedings of the Computer Graphics and Applications, 12th Pacific Conference*, (Washington, DC, USA), pp. 298–305, IEEE Computer Society, 2004.

[97] H. Wang, S. Yan, T. Huang, and X. Tang, "Maximum unfolded embedding: formulation, solution, and application for image clustering," in *MULTIME-*

DIA '06: Proceedings of the 14th annual ACM international conference on Multimedia, (New York, NY, USA), pp. 45–48, ACM, 2006.

[98] X. He, "Incremental semi-supervised subspace learning for image retrieval," in *MULTIMEDIA '04: Proceedings of the 12th annual ACM international conference on Multimedia*, (New York, NY, USA), pp. 2–8, ACM, 2004.

[99] H. Sahbi, P. Etyngier, J.-Y. Audibert, and R. Keriven, "Manifold learning using robust graph laplacian for interactive image search," in *Computer Vision and Pattern Recognition (CVPR)*, pp. 1–8, June 2008.

[100] H. Zhang, O. van Kaick, and R. Dyer, "Spectral methods for mesh processing and analysis," in *Proc. of Eurographics State-of-the-art Report*, pp. 1–22, 2007.

[101] M. Belkin and P. Niyogi, "Laplacian eigenmaps for dimensionality reduction and data representation," *Neural Comput.*, vol. 15, no. 6, pp. 1373–1396, 2003.

[102] D. L. Donoho and C. Grimes, "Hessian eigenmaps: Locally linear embedding techniques for high-dimensional data," *PNAS*, vol. 100, pp. 5591–5596, May 2003.

[103] J. Ham, D. D. Lee, S. Mika, and B. Schölkopf, "A kernel view of the dimensionality reduction of manifolds," in *ICML '04: Proceedings of the twenty-first international conference on Machine learning*, (New York, NY, USA), p. 47, ACM, 2004.

[104] C. Fowlkes, S. Belongie, F. Chung, and J. Malik, "Spectral grouping using the nystrom method," *IEEE Transactions on Pattern Analysis and Machine Intelligence*, vol. 26, pp. 214–225, Feb. 2004.

[105] V. Jain and H. Zhang, "Robust 3D shape correspondence in the spectral domain," in *SMI '06: Proceedings of the IEEE International Conference on Shape Modeling and Applications 2006 (SMI'06)*, (Washington, DC, USA), p. 19, IEEE Computer Society, 2006.

[106] F. L. Bookstein, "Principal warps: Thin-plate splines and the decomposition of deformations," *IEEE Transactions on Pattern Analysis and Machine Intelligence*, vol. 11, no. 6, pp. 567–585, 1989.

[107] M. Cardle, M. Vlachos, S. Brooks, E. Keogh, and D. Gunopulos, "Fast motion capture matching with replicated motion editing," in *Proceedings of SIGGRAPH 2003 - Sketches and Applications*, July 2003.

[108] F. Liu, Y. Zhuang, F. Wu, and Y. Pan, "3D motion retrieval with motion index tree," *Computer Vision and Image Understanding*, vol. 92, no. 2-3, pp. 265–284, 2003.

[109] C.-Y. Chiu, S.-P. Chao, M.-Y. Wu, S.-N. Yang, and H.-C. Lin, "Content-based retrieval for human motion data," *Journal of Visual Communication and Image Representation, Special Issue on Multimedia Database Management Systems*, vol. 15, no. 3, pp. 446–466, 2004.

[110] K. Forbes and E. Fiume, "An efficient search algorithm for motion data using weighted pca," in *SCA'05: Proceedings of the 2005 ACM SIGGRAPH/Eurographics symposium on Computer animation*, (New York, NY, USA), pp. 67–76, ACM Press, 2005.

[111] C. Faloutsos, M. Ranganathan, and Y. Manolopoulos, "Fast subsequence matching in time-series databases," in *SIGMOD'94: Proceedings of the 1994 ACM SIGMOD international conference on Management of data*, (New York, NY, USA), pp. 419–429, ACM Press, 1994.

[112] R. Agrawal, C. Faloutsos, and A. N. Swami, "Efficient similarity search in sequence databases," in *FODO'93: Proceedings of the 4th International Conference on Foundations of Data Organization and Algorithms*, (London, UK), pp. 69–84, Springer-Verlag, 1993.

[113] K.-P. Chan and A. W.-c. Fu, "Efficient time series matching by wavelets," in *Proceedings of the 15th International Conference on Data Engineering(ICDE)*, (Washington, DC, USA), p. 126, IEEE Computer Society, 1999.

[114] K. Chakrabarti, E. Keogh, S. Mehrotra, and M. Pazzani, "Locally adaptive dimensionality reduction for indexing large time series databases," *ACM Trans. Database Syst.*, vol. 27, no. 2, pp. 188–228, 2002.

[115] R. Dannenberg, W. Birmingham, G. Tzanetakis, C. Meek, N. Hu, and B. Pardio, "The musart testbed for query-by-humming evaluation," in *Int. Conf. on Music Information Retrieval (ISMIR)*, 2003.

[116] M. Müller, T. Röder, and M. Clausen, "Efficient content-based retrieval of motion capture data," in *ACM SIGGRAPH*, 2005.

[117] M. Müller, T. Röder, and M. Clausen, "Efficient indexing and retrieval of motion capture data based on adaptive segmentation," in *the 4th Intl. Workshop on Content-Based Multimedia Indexing (CBMI 2005)*, 2005.

[118] M. Müller and T. Röder, "Motion templates for automatic classification and retrieval of motion capture data," in *ACM SIGGRAPH / Eurographics SCA*, 2006.

[119] B. Demuth, T. Röder, M. Müller, and B. Eberhardt, "An information retrieval system for motion capture data," in *the 28th European Conference on Information Retrieval (ECIR 2006)* (M. Lalmas, ed.), pp. 373–384, LNCS 3936, 2006.

[120] Y. Lin, "Efficient human motion retrieval in large databases," in *GRAPHITE'06: Proceedings of the 4th international conference on Computer graphics and interactive techniques in Australasia and Southeast Asia*, (New York, NY, USA), pp. 31–37, ACM Press, 2006.

[121] D. H. Douglas and T. K. Peucker, "Algorithms for the reduction of the number of points required to represent a digitized line or its caricature," *Canadian Cartographer*, vol. 10, pp. 112–122, 1973.

[122] H. Sakoe and S. Chiba, "Dynamic programming algorithm optimization for spoken word recognition," *IEEE Transactions on Acoustics, Speech, and Signal Processing*, vol. 26, pp. 43–49, Feb. 1978.

[123] "Graphics lab, motion capture database, carnegie mellon university." http://mocap.cs.cmu.edu/.

[124] E. J. Keogh, S. Chu, D. Hart, and M. J. Pazzani, "An online algorithm for segmenting time series," in *ICDM '01: Proceedings of the 2001 IEEE International Conference on Data Mining*, (Washington, DC, USA), pp. 289–296, IEEE Computer Society, 2001.

[125] S. Katz, G. Leifman, and A. Tal, "Mesh segmentation using feature point and core extraction," *The Visual Computer*, vol. 21, pp. 649–658, Sept. 2005.

[126] E. Rabani and S. Toledo, "Out-of-core SVD and QR decompositions," in *In Proceedings of the 10th SIAM Conference on Parallel Processing for Scientific Computing*, 2001.

[127] M. Wichterich, I. Assent, P. Kranen, and T. Seidl, "Efficient emd-based similarity search in multimedia databases via flexible dimensionality reduction," in *SIGMOD '08: Proceedings of the 2008 ACM SIGMOD international conference on Management of data*, (New York, NY, USA), pp. 199–212, ACM, 2008.

[128] H. Zhang, A. Sheffer, D. Cohen-Or, Q. Zhou, O. van Kaick, and A. Tagliasacchi, "Deformation-drive shape correspondence," *Computer Graphics Forum (Special Issue of Symposium on Geometry Processing 2008)*, vol. 27, no. 5, pp. 1431–1439, 2008.

[129] O. Duchenne, J.-Y. Audibert, R. Keriven, J. Ponce, and F. Segonne, "Segmentation by transduction," *Computer Vision and Pattern Recognition, IEEE Computer Society Conference on*, vol. 0, pp. 1–8, 2008.

[130] C. G. M. Snoek and M. Worring, "Multimodal video indexing: A review of the state-of-the-art," *Multimedia Tools and Applications*, vol. 25, pp. 5–35, Jan. 2005.

[131] P. S. Dhillon, S. Nowozin, and C. H. Lampert, "Combining appearance and motion for human action classification in videos," *Computer Vision and Pattern Recognition Workshop*, vol. 0, pp. 22–29, 2009.

[132] S. Johnson and M. Everingham, "Combining discriminative appearance and segmentation cues for articulated human pose estimation," in *2nd IEEE International Workshop on Machine Learning for Vision-based Motion Analysis (MLVMA09)*, 2009.

[133] X. Zhang, G. Fan, and L.-S. Chou, "Two-layer generative models for estimating unknown gait kinematics," in *2nd IEEE International Workshop on Machine Learning for Vision-based Motion Analysis (MLVMA09)*, 2009.

[134] R. Bodor, A. Drenner, D. Fehr, O. Masoud, and N. Papanikolopoulos, "View-independent human motion classification using image-based reconstruction," *Image and Vision Computing*, vol. 27, no. 8, pp. 1194 – 1206, 2009.

Appendix - Vitae

Gary K.L. Tam received a first-class honors B.Eng. degree in Computer Science from the Hong Kong University of Science and Technology in 2002. In the same year, Gary Tam was selected as one of the 23 Youth Telecom Ambassadors in the Event of "ITU Telecom Asia 2002 Youth Forum" in Hong Kong. In 2004, He received an MPhil degree from City University of Hong Kong. After graduation, he worked as an instructor in the Department of Computer Science of the same university for about two years. He is currently a PhD student at the University of Durham. His research mainly focuses on geometry processing, multimedia retrieval and computer graphics.

Publications

1. Gary Tam and Rynson Lau, "Embedding Retrieval of 3D Articulated Geometry Models," currently under review, submitted to ACM Multimedia 2009.

2. Gary Tam, Rynson Lau and Jianmin Zhao, "A 3D Geometry Search Engine in Support of Learning," International Journal of Distance Education Technologies (JDET), to appear.

3. Gary Tam, Qingzheng Zheng, Mark Corbyn and Rynson Lau, "Motion Retrieval Based on Energy Morphing," Proc. IEEE International Symposium on Multimedia, 2007.

4. Gary Tam and Rynson Lau, "Deformable Model Retrieval Based on Topological and Geometric Signatures," IEEE Transactions on Visualization and Computer Graphics, vol. 13, no. 3, pp. 470-482, 2007.

5. Hui Xiao, Gary Tam, Frederick Li, and Rynson Lau, "Dynamic Radiosity on Multi-resolution Models," Proceedings of International Conference on Artificial Reality and Telexistence (ICAT), LNCS 4282, pp. 753-763, Springer-Verlag, Nov. 2006.

6. Hui Xiao, Rynson Lau and Gary Tam, "An Efficient Illumination Framework for Web-based Lighting Engineering Education," Proceedings of International Conference on Web-based Learning (ICWL), pp. 105-111, July 2006.

7. Gary Tam, "Matching and Retrieval of 3D Deformable Models," Thesis for the degree of Master of Philosophy, City University of Hong Kong, Nov 2004.

8. Gary Tam, Rynson Lau, and C.W. Ngo, "Deformable Geometry Model Matching by Topological and Geometric Signatures," Proceedings of International Conference on Pattern Recognition (ICPR), pp. 910-913, Aug. 2004.

9. Gary Tam, Rynson Lau, and C.W. Ngo, "Deformable Geometry Model Matching Using Bipartite Graph," Proceedings of Computer Graphics International (CGI), pp. 335-342, June 2004.